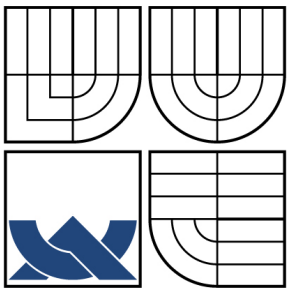
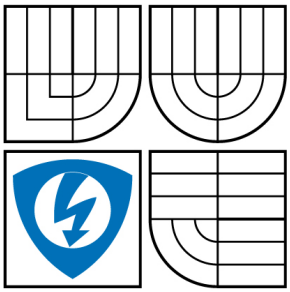


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍDICÍ SYSTÉM MOBILNÍHO ROBOTU MINIDARPA

MINIDARPA ROBOT - SOFTWARE DESIGN

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

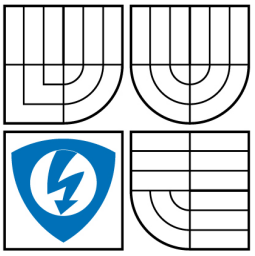
AUTOR PRÁCE — Bc. MARTIN KOPECKÝ

AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ KOPEČNÝ, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Martin Kopecký	***ID:***	78525
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Řídicí systém mobilního robotu Minidarpa

POKYNY PRO VYPRACOVÁNÍ:

Zpracujte algoritmus pro plánování trajektorie mobilního robotu Minidarpa.

DOPORUČENÁ LITERATURA:

ŽALUD, L., KOPEČNÝ, L. Teleoperated Reconnaissance Robotic System. In SSRR 2004 - IEEE International Workshop on Safety, Security, and Rescue Robotics. Bonn - Germany, Fraunhofer-Gesellschaft. 2004. p. 1 - 6.

Termín zadání:	8.2.2010	***Termín odevzdání:***	24.5.2010

Vedoucí práce: Ing. Lukáš Kopečný, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

Abstrakt

Tato diplomová práce se zabývá tematikou autonomních mobilních robotů. V návaznosti na předchozí bakalářskou práci je cílem návrh řídicího systému venkovního mobilního robotu pro soutěž typu Minidarpa. Jeho praktickým využitím by mohl být např. autonomní převoz materiálu. Při návrhu řídicího systému je pozornost soustředěna především na plánovací algoritmy, které jsou rozděleny na lokální a globální. Jedním z těžišť práce je globální plánování. To za pomoci grafových algoritmů plánuje body, jež je třeba projet k dosažení požadovaného cíle. Globální mapa je definována pomocí RNDF souboru, ten je však nutno pro potřeby plánování předzpracovat. Dále jsou v práci navrženy třídy v jazyce C++ pro komunikaci a obsluhu senzorických subsystémů. Řídicí systém zahrnuje také grafické rozhraní pro obsluhu robotu a možnost dálkového ovládání pomocí bezdrátového gamepadu.

Klíčová slova

mobilní robot, řídicí systém, Robotour, Minidarpa, plánovací algoritmus, RNDF

Abstract

This master‘s thesis deals with theme of autonomous mobile robots. In sequence of the bachelor’s thesis its goal is to design a control system of the outdoor mobile robot for Minidarpa type competition. It could be practically used for an autonomous transportation of material. In design of the control system the attention is aimed at planning algorithms which are divided into the local and the global parts. Most of the work is paid to a global planning algorithm, which plans, in help with graph algorithms, points needed to travel to achieve the goal. The global map is defined in RNDF file which must be preprocessed for needs of planning. There are designed classes in C++ language for communication with subsystems too. The control system further contains graphical interface for operation of the robot and a remote control through a wireless gamepad.

Keywords

mobile robot, control system, Robotour, Minidarpa, planning algorithm, RNDF

Bibliografická citace

KOPECKÝ, M. *Řídicí systém mobilního robotu Minidarpa*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 63 s. Vedoucí diplomové práce Ing. Lukáš Kopečný, Ph.D.

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma Řídicí systém mobilního robotu Minidarpa jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : ……

podpis autora

P o ď k o v á n í

Děkuji tímto vedoucímu práce Ing. Lukáši Kopečnému, Ph.D. a doc. Ing. Luďku Žaludovi, Ph.D. za cenné připomínky a rady při vypracování mé diplomové práce.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

CS	Central stop
DARPA	Defense Advanced Research Projects Agency
DGPS	Differential Global Positioning System
GPS	Global Positioning Systém
GUI	Graphical User Interface
MDF	Mission Data File
RNDF	Route Network Definition File

1. ÚVOD

Robotika je v současné době velmi moderní a rozvíjející se odvětví. Vedle stacionárních robotů, dnes již hojně využívaných v průmyslových aplikacích, existují roboty mobilní, které jsou zatím v běžném životě méně časté. Praktické nasazení zejména autonomních mobilních robotů omezují mnohé, dosud uspokojivě nevyřešené problémy, mezi něž patří např. přesná a spolehlivá lokalizace v neznámém prostředí, robustní plánovací algoritmy atd.

Tato diplomová práce navazuje na předchozí bakalářskou práci [7], jejímž cílem byl návrh řídicího systému mobilního robotu pro soutěž typu Minidarpa. V České republice se takovýto druh soutěže pořádá pod názvem Robotour [3]. Obě soutěže jsou inspirovány soutěží DARPA Grand Challenge [11], kde se utkávají velké automobily v autonomní jízdě po zadané trase s požadavkem chovat se jako člověkem řízené auto. To znamená především držet se na silnici ve svém pruhu a reagovat na případné překážky, popř. dodržovat dopravní předpisy. Na rozdíl od ní ale v našich podmínkách soutěží pouze malé roboty v jízdě po parkových cestách, avšak s podobnými požadavky. Ve zmíněné bakalářské práci byl navržen řídicí systém především po hardwarové stránce a velice jednoduché algoritmy pro komunikaci se subsystémy. Navrhovaný řídicí systém je součástí kolektivní práce, která dále zahrnuje senzorické a pohonné subsystémy daného robotu. Tyto subsystémy zpracovávají týmoví kolegové, ovšem v některých částech této práce jsou určité jejich aspekty naznačeny v rozsahu nezbytném pro popsání řídicího systému jako celku.

V této práci je původně navržený řídicí systém přepracován a doplněn o nové algoritmy a funkce. Popisovaný mobilní robot se již zúčastnil několika ročníků soutěže Robotour, z čehož vyplynuly požadavky na úpravu řídicího systému, především plánovacích algoritmů, proto je na ně v práci soustředěna pozornost. K úspěšnému plánování jsou však potřeba především informace ze senzorů, proto návrh rovněž zahrnuje implementací rozhraní pro jednoduchou komunikaci s různými senzorickými subsystémy. Ze senzorických subsystémů je pozornost věnována spolupráci s kamerovým subsystémem, který je pro navigaci robotu,

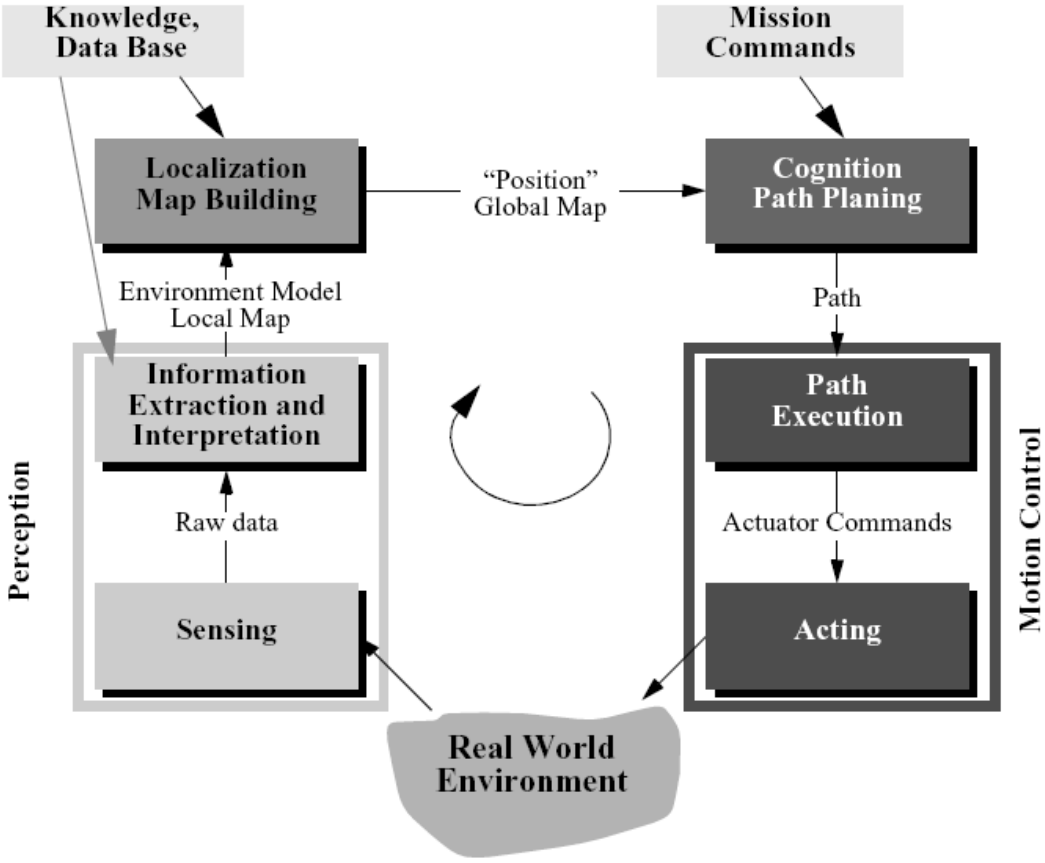
zejména ve venkovním prostředí, velmi důležitý. Samozřejmě jsou rozpracovány všechny části potřebné pro zajištění funkčnosti řídicího systému

Práce je rozdělena na dvě hlavní části. První se zabývá rešerší v dané oblasti mobilní robotiky především z hlediska řídicích systémů a plánování. Ve druhé části je popsán navržený řídicí systém společně s komunikačními třídami a jejich softwarová implementace.

2. TEORIE

2.1 INTELIGENTNÍ MOBILNÍ ROBOT

Na obrázku 2.1 je znázorněno obecné schéma inteligentního robotického systému dle literatury [8]. Jedná se o jeden z mnoha různých přístupů, ale je názorné a pro účely této práce vhodné, neboť ho lze aplikovat jak na mobilní, tak i na stacionární roboty a jsou na něm vidět všechny základní prvky uvedeného systému. V našem případě nás toto schéma bude zajímat především z pohledu mobilních robotů.



Obrázek 2.1: Obecné schéma inteligentního robotického systému [8]

Blok snímání („Sensing“) zajišťuje sběr dat z reálného prostředí, v tomto případě to mohou být např. vzdálenosti překážek měřené pomocí ultrazvukových či laserových snímačů, obraz okolí snímaný kamerou (klasická kamera, 3D, IR, …),

teplota, ujetá vzdálenost atd. Získané informace je třeba zpracovat tak, aby byly srozumitelné pro počítačový systém, který ve většině případů robot řídí. To zajišťuje následující blok („Information Extraction and Interpolation“) tvořící dohromady s předchozím blokem část nazvanou vnímání. Výstupem může být např. model prostředí, či nějaký druh mapy, ve které se robot dokáže orientovat.

Robot by měl být schopen se v modelovaném prostředí nějakým způsobem orientovat, popř. mapu doplňovat o nově získané poznatky, což zajišťuje blok lokalizace („Lokalization, Map Building“). Výstupem tohoto bloku je znalost pozice robotu v globální mapě.

Jak pro výše zmíněné zpracování vstupních dat, tak i pro lokalizaci robotu jsou zapotřebí jisté znalosti, které obecně slouží k transformaci dat na informace, což samozřejmě platí i v tomto případě. Znalosti mohou být ve vhodně zakódované formě uloženy v bázi znalostí („Knowledge, Data Base“).

Jakmile zná robot svoji pozici, přichází na řadu plánování, což tvoří nedílnou součást inteligence robotu. Vstupem této části je požadovaný cíl pohybu, resp. úkol. Plánovat je možno na více úrovních abstrakce a výstup tvoří trasa, či posloupnost úkonů, které robot musí vykonat pro splnění požadovaného cíle.

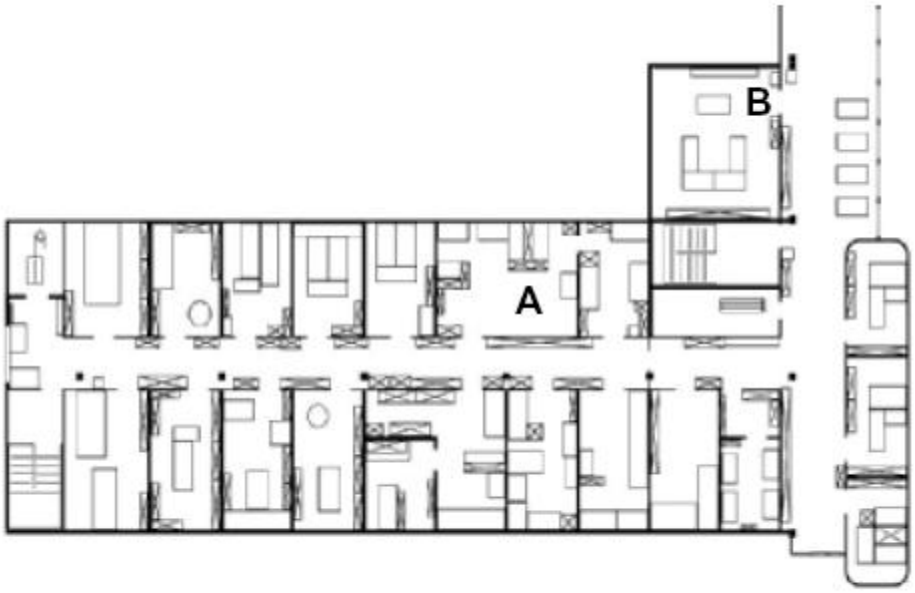
Za realizování těchto plánů zodpovídá poslední blok, tzv. řízení pohybu (také se někdy nazývá akční částí – „Motion Control“). Tento blok se skládá ze dvou částí. První, která se stará o převod požadované trasy na pokyny pro řízení efektorů, v našem případě motorů, druhá zajišťuje přímé působení na pohon robotu, např. řídicí jednotka motoru. [8]

2.2 STRUKTURA ŘÍDICÍCH SYSTÉMŮ MOBILNÍCH ROBOTŮ

2.2.1 Úvod

K návrhu řídicího systému autonomního mobilního robotu lze přistoupit z více různých hledisek. Mezi nejjednodušší systémy patří dva základní. Prvním je řídicí systém založený pouze na předem definovaných pravidlech a reakcích na nějaké vnější podněty, např. jeď rovně a při detekci překážky se otoč. Druhou možností je systém založený na lokalizaci a s ní spojenou tvorbou mapy prostředí.

Rozdíl mezi oběma přístupy je možno přiblížit na příkladu prostředí z obrázku 2.2. Pokud má robot za úkol dostat se z místnosti A do místnosti B, může se zdát, že se bez mapy prostředí neobejde. Tento úkol ovšem může zvládnout i se systémem založeným pouze na reakcích na vnější podněty. Pokud bude mít robot naprogramováno například pouze pravidlo „sleduj levou zeď“, může se do místnosti B bez problémů dostat. Je ale samozřejmé, že takovýto pohyb nebude optimální ani z časového pohledu ani z hlediska ujeté dráhy. Bude však ještě zapotřebí nějak rozpoznat, že se do místnosti B už dostal – např. podle barvy koberce.



Obrázek 2.2: Příklad prostředí pro pohyb mobilního robotu [8]

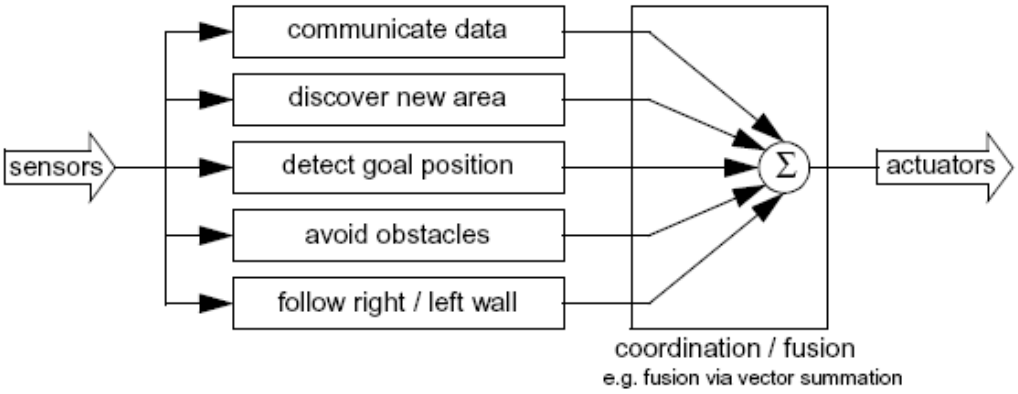
Ke složitějším a inteligentnějším systémům se řadí především tzv. plánování na vyšší úrovni (např. plánovací algoritmus STRIPS), popř. systémy založené na neuronových sítích či genetických algoritmech. Existují samozřejmě různé kombinace výše zmíněných přístupů, tvořící skupinu tzv. hybridních systémů. [8][9]

2.2.2 Reaktivní řídicí systém

Architektura tohoto přístupu je vidět na obrázku 2.3. Její inspirací je způsob chování zvířat. Principem je přímé ovládání akčních částí robotu (efektorů) systémy, které zpracovávají údaje ze senzorů podle předem definovaných schémat.

Hlavní výhodou reaktivního systému je jednoduchá a velmi rychlá implementace řídicích algoritmů. Dále není nutno mít přímo vyjádřený cíl a také je tato architektura schopna velmi rychle reagovat na vnější podněty.

Samozřejmě má tento systém několik nevýhod – především je to jeho jednoúčelnost, pro aplikaci takového řídicího systému v jiném prostředí bude totiž nutná velmi složitá úprava algoritmů. Dalším nedostatkem, např. při naznačeném sledování zdi, je časová náročnost – robot neví, zda se přibližuje k cíli, či nikoliv – a závislost na konkrétních podmínkách daného prostředí. Třetí velkou nevýhodu představuje obtížnější fúze rozdílných naprogramovaných reakcí – např. sledování zdi a objíždění překážky. Pokud nastanou požadavky pro reakci na oba podněty současně, může být pro řídicí systém velmi těžké rozhodnout, který z nich má větší prioritu z hlediska splnění cíle mise.



Obrázek 2.3: Reaktivní architektura řídicího systému [8]

Tento typ řídicího systému je možno dále rozdělit dle způsobu implementace dané architektury.

2.2.2.1 Architektura podřízenosti

Jde o nejčastěji používanou architekturu těchto řídicích systémů, založenou na hierarchickém rozdělení jednotlivých vrstev, z nichž každá reprezentuje jeden typ chování. Každé vrstvě je přidělena priorita od nejvyšší priority pro nejnižší vrstvu – např. detekce překážek, až po nejnižší prioritu pro nejvyšší vrstvu – např. kontrola vykonávání zadaných cílů. Nevýhodou této architektury je nedeterminističnost

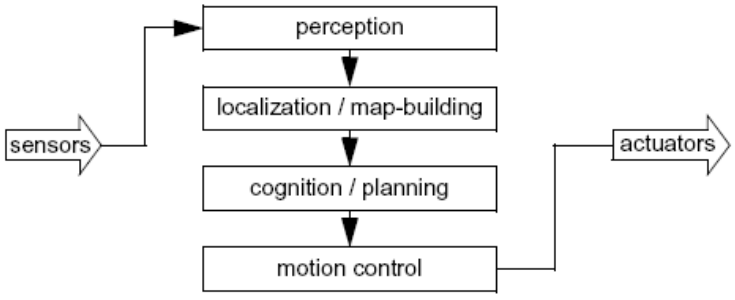
chování robotu, protože každá vrstva je spouštěna příchodem nové události od snímačů, a nelze tedy dopředu odhadnout přesné chování robotu.

2.2.2.2 Architektura vzorců chování

Tato architektura je založena na tzv. vzorcích chování, kterými jsou v podstatě předem definované reakce na vnější podněty. Výstupem každého vzorce je rychlost a směr pohybu robotu jako reakce na daný podnět. Takovýmto vzorcem může být např. „dopředu“, „zabránění kolizi“, „po cestě“ atd. Výsledný pohyb robotu je na rozdíl od architektury podřízenosti, kde byl pohyb vždy určován pouze jednou vrstvou, vektorovým součtem všech navrhovaných pohybů. Nevýhodou je přesto opět nedeterminističnost pohybu robotu. [8][9]

2.2.3 Řídicí systém založený na mapování prostředí

Protikladem reaktivního řídicího systému je systém založený na mapování a lokalizaci – někdy také nazývaný systém založený na funkční dekompozici. Jeho architektura je zachycena na obrázku 2.4. Principem je sekvenční řazení jednotlivých bloků (subsystémů), zajišťujících jednotlivé funkce systému. Tok dat těmito bloky probíhá v cyklu snímání-plánování-akce, respektive snímání-modelování-plánování-akce. Při tomto přístupu robot v každém okamžiku relativně přesně zná svoji absolutní polohu v mapě prostředí. Aby toho bylo možno dosáhnout, je však nutné zabývat se i problematikou lokalizace a plánování, s nimi související problematikou reprezentací nepřesnosti snímání daného prostředí snímači robotu atd.



Obrázek 2.4: Architektura řídicího systému založeného na navigaci v mapě [8]

Toto řešení přináší ovšem značné výhody oproti předchozímu přístupu. Mezi přednosti patří zvláště znalost polohy robotu v každém okamžiku, což může být

užitečné i pro případného operátora, který by dohlížel na činnost robotu. Přínosná je i samotná existence mapy prostředí, která může být použita jako komunikační prostředek mezi operátorem a robotem, kupříkladu při zadávání požadovaných cílů prostřednictvím této mapy. A samozřejmě je výhodou snadná přenositelnost řídicího systému do jiného prostředí, neboť stačí pouze vyměnit mapu prostředí a robot může pokračovat v činnosti.

Problémem při tomto přístupu je ovšem vyšší složitost řídicího systému i robotu samotného, především z hlediska nároků na snímače, protože při jejich nefunkčnosti nebo poskytnutí chybných údajů může selhat celý řídicí systém. Dalším nedostatkem je, že pokud je řídicí systém založen pouze na mapovém principu, může být slabým místem celého systému mapa samotná. Bude-li nepřesná či špatná, může se stát, že bude robot „ztracen“. V neposlední řadě je nevýhodou pomalá reakce na některé kritické příchozí události, např. na nečekanou pohyblivou překážku před robotem.

Koordinace jednotlivých modulů může být řešena buď hierarchicky, nebo pomocí tzv. tabule, kde tabule představuje určité místo, na němž si jednotlivé moduly vyměňují informace. Kupříkladu modul vnímání pak zapisuje na určené místo informace od senzorů, modul plánování aktuální plán činností robotu, na druhé straně si např. realizátor plánů vybere z tabule plán, který je bezkonfliktní s aktuálně naměřenými informacemi od snímačů. Systém založený na funkční dekompozici je možno rozdělit do několika vrstev. Běžně je užíván standard společnosti NASA/NBS [1] , který je určen především pro telerobotické aplikace, ale lze jej použít i pro autonomní roboty. Uvedený standard dělí systémy (jak hierarchické tak i systémy využívající tabuli) na šest základních vrstev:

1. Převod souřadnic („Coordinate transform“) – transformace informací od senzorů do souřadného systému robotu a ovládání akcí robotu.
2. Prvotní vrstva („Primitive“) – dynamika systému a ovládání rychlosti pohybu.
3. Bezpečný pohyb („E-move“) – detekce a vyhýbání se překážkám.
4. Příkazová vrstva („Task“) – realizace plánů do posloupnosti příkazů.
5. Plánovací vrstva („Services Bay“) – plánování činností.

6. Úkolová vrstva („Service Mission“) – rozhodování o proveditelnosti cílového úkolu.

[8][9] [1]

2.2.4 Hybridní řídicí systém

V praxi jsou řídicí systémy mobilních robotů většinou tvořeny kombinací uvedených přístupů. Obvyklý řídicí systém je rozdělen na vrstvy, podobně jako u systémů založených na funkční dekompozici – mapování. Jednotlivé vrstvy jsou však tvořeny různými architekturami podle účelu dané vrstvy. Vrstvy na nízké úrovni, zajišťující rychlé reakce na podněty, jsou zpravidla tvořeny reaktivními systémy, zatímco plánování či modelování prostředí je opět tvořeno systémem založeným na funkční dekompozici. Pro nejvyšší vrstvy plánování či zadávání cílů operátorem je výhodné použít např. systémy vyššího abstraktního plánování. [8] [9]

2.2.5 Shrnutí

Jako nejlepší v praxi využitelný se dle výše popsaných principů a vlastností jeví hybridní řídicí systém, realizovaný spojením mapového a reaktivního systému. Jeho největší předností je možnost kombinovat různé funkce z obou posuzovaných systémů, jako např. rychlá reakce na vnější podměty z reaktivního systému a tvorba mapy a znalost polohy z mapového systému.

2.3 PROBLEMATIKA TVORBY MAP

Problém reprezentace prostředí, ve kterém se robot pohybuje, je v přímém vztahu s přesností lokalizace robotu. Při tvorbě mapové reprezentace prostředí musí být brány v úvahu tyto tři základní vztahy:

1. Přesnost mapy musí odpovídat přesnosti, s jakou robot chce dosáhnout požadovaného cíle.
2. Přesnost mapy a její typ musí být v souladu s přesností senzorů, které robot má k dispozici.
3. Komplexnost a propracovanost mapy má přímý vliv na složitost navigace, lokalizace a ostatních systémů spojených s mapou. [8]

Jak také vyplývá z předchozích vztahů, je samozřejmostí při tvorbě map všudypřítomnost nejistot a šumů, které především zkreslují měření snímačů. Je proto nutné při návrhu reprezentace prostředí na tuto skutečnost brát ohled.

Pro potřeby plánování lze použít více vrstev abstrakcí map. Například pro plánovací algoritmus rozdělený na lokální a globální část, jako tomu je v této práci, lze použít dvě vrstvy map. Pro lokální plánování např. mřížky obsazenosti a pro globální plánování např. orientované grafy. I přesto, že jejich využití při lokálním plánování je výhodné, byl stávající princip lokálního plánování zachován. Pozornost byla zaměřena především na globální plánování, z důvodů požadavků pravidel soutěže Robotour. Detailněji tedy budou rozebrány mapy reprezentované orientovanými grafy, protože jich bude využito v globálním plánovacím modulu navrhovaného řídicího systému.

Obecně je tato problematika velmi rozsáhlá a je zmíněna především z důvodu úzké souvislosti s tématem této práce, protože správný výběr reprezentace prostředí přímo ovlivňuje složitost následného plánování a lokalizace.

2.3.1 Mřížky obsazenosti

Mřížky obsazenosti je možno použít k reprezentaci map pro lokální plánování. Lze v nich interpretovat různé překážky pomocí fúze dat z různých typů senzorů. Tu je možno v mřížkách obsazenosti velmi jednoduše realizovat. Ani následné plánování optimální trasy pak není obtížné.

Princip spočívá v tom, že prostředí, v němž se robot pohybuje, je rozděleno na stejně velké buňky. Každá z nich má přiřazenu číselnou hodnotu reprezentující nějakou vlastnost dané buňky, např. její obsazenost či neobsazenost překážkou. V jiné modifikaci – tzv. potenciálových polích – může hodnota buňky vyjadřovat vzdálenost k cíli. Plánování poté probíhá buď přes neobsazené buňky, nebo například po buňkách se snižující se hodnotou (pokud má cíl nejnižší hodnotu).

Jako nejvhodnější se pro zadanou problematiku jeví pravděpodobnostní mřížky obsazenosti, v nichž lze poměrně jednoduše interpretovat nejistoty používaných senzorů při detekci překážek a také je možno integrovat pomocí zmíněné fúze dat do těchto mřížek informace z různých typů senzorů, např. z ultrazvukových snímačů, laserového skeneru či kamery. [4][12]

2.4 REPREZENTACE MAPY POMOCÍ GRAFU

2.4.1 Úvod

Pro reprezentaci mapy na vyšší úrovni – tedy pro globální plánování – lze s výhodou použít grafů, resp. orientovaných grafů. Grafem je v tomto pojetí myšlena dvojice množin vrcholů a hran. Pro reprezentaci mapy je možno jako vrcholy grafu použít určitá specifická místa na mapě, jako jsou např. křižovatky. Hranami daného grafu jsou poté cesty spojující jednotlivé křižovatky. Výhodou této reprezentace je snadné plánování v takovéto mapě, protože existuje velké množství algoritmů pro hledání a různé jiné operace s grafy, které většinou nejsou příliš výpočetně náročné. Pro plánování je však třeba graf doplnit informacemi o vzdálenosti mezi jednotlivými uzly, či nějaké další potřebné informace. Tím vzniká tzv. ohodnocený graf, který už dostatečně dobře reprezentuje globální mapu prostředí. Dalším nutným parametrem je informace o možnosti směru průjezdu jednotlivými hranami, což upravuje graf do podoby, kterou lze využít pro plánování, na tzv. orientovaný ohodnocený graf.

Dalším aspektem je způsob uložení grafu v paměti řídicího systému tak, aby bylo jeho prohledávání co nejjednodušší. V následujícím popisu budou uvedeny některé možné způsoby a jejich vlastnosti. Každá reprezentace má své výhody i nevýhody, je tedy třeba zvolit takovou, která vyhovuje požadované situaci. V tomto případě budou reprezentace posuzovány především podle vhodnosti pro plánování, tedy prohledávání grafu.

2.4.2 Seznam hran

Graf je uložen jako seznam všech hran, tj. dvojic vrcholů (např. UV, VU, …). Tato reprezentace je vhodná spíše pro zadávání grafů, než pro jejich ukládání v počítači a následnou práci s nimi.

2.4.3 Matice sousednosti

V této matici jsou uloženy informace, které vrcholy spolu sousedí, resp. pro každou dvojici vrcholů je v matici uloženo, zda vede hrana mezi nimi. Tuto reprezentaci lze použít pro orientované grafy. Výhodou je jednoduchá práce s maticí

a také možnost reprezentace ohodnocení hran. Pokud je v matici uloženo ohodnocení vzdálenostmi, jedná se o tzv. matici vzdáleností.

2.4.4 Seznam sousedů

Princip spočívá v ukládání seznamu sousedních vrcholů pro každý jednotlivý vrchol. Opět vhodná reprezentace pro orientované grafy. [2]

2.4.5 Shrnutí

Jako nejvhodnější se pro použití k plánování trasy v tomto případě jeví reprezentace grafu pomocí matice sousednosti, především z důvodu jednoduché implementace a snadné reprezentace ohodnocení grafu.

2.5 HLEDÁNÍ OPTIMÁLNÍ TRASY V ORIENTOVANÉM GRAFU

2.5.1 Úvod

V mobilní robotice je hledání optimální trasy při plánování robotu velice důležité. Trasa může být optimální z mnoha hledisek. Nejčastěji ji posuzujeme podle délky ujeté vzdálenosti, ale dalším hlediskem může být třeba složitost terénu, ve kterém se robot pohybuje. Pokud máme například malý mobilní robot, stává se pro něj průjezd těžkým terénem velmi obtížný, proto je třeba hledat takovou trasu, která bude pro daného robota optimální vzhledem ke složitosti terénu. Z těchto požadavků vyplývá, že pokud máme prostředí reprezentované pomocí orientovaného grafu, je vhodné každé spojnici vrcholů přidělit nějaké parametry. Těmito parametry může být například vzdálenost vrcholů, obtížnost terénu atd. Poté o takovém grafu mluvíme jako o ohodnoceném orientovaném grafu.

Pro nalezení požadované trasy v takovémto grafu existuje řada algoritmů. V této práci budou popsány dva z nich a to Dijkstrův a Floyd-Warshallův algoritmus. Vybrány byly především pro jejich jednoduchou implementaci a použitelnost. [2]

2.5.2 Dijkstrův algoritmus

Najde nejkratší cestu z jednoho vrcholu grafu do jiného. Princip spočívá v postupném procházení všech vrcholů směrem z výchozího a ukládání vždy nejbližšího vrcholu. Ohodnocení grafu musí být v případě tohoto algoritmu pouze nezáporné, což však v případě ohodnocení délkami hran není problémem, protože

délka je vždy nezáporná. Výhodou tohoto algoritmu je jeho konečnost. Tedy pro konečný počet uzlů najde nejkratší trasu vždy v konečném počtu kroků. Výsledkem jsou dvě matice. Jedna obsahuje nejkratší vzdálenosti mezi jednotlivými vrcholy, druhá informace potřebné pro rekonstrukci dané nejkratší cesty.

2.5.3 Floyd-Warshallův algoritmus

Najde nejkratší vzdálenosti mezi všemi vrcholy daného grafu. Výhodou je okamžitá znalost všech vzdáleností, nevýhodou složitější rekonstrukce nejkratších cest. Výsledkem jsou opět dvě matice, podobně jako u Dijkstrova algoritmu. Jedna obsahuje nejkratší vzdálenosti mezi jednotlivými vrcholy, druhá informace potřebné pro rekonstrukci dané nejkratší cesty [2].

2.5.4 Shrnutí

Jako nejvhodnější pro zadanou problematiku plánování optimální trasy byl zvolen Dijkstrův algoritmus, pro jeho jednoduchou implementaci a snadnou rekonstrukci optimální trasy. Existuje sice celá řada modifikací tohoto algoritmu z hlediska výpočetní rychlosti a náročnosti, v tomto případě však byla zvolena, z důvodu dostatku výpočetního výkonu, základní implementace.

Ve výše uvedeném popisu se hovoří o hledání nejkratší trasy, pouhým zaměněním ohodnocení grafu za jiný parametr než vzdálenost lze dosáhnout hledání optimální trasy z jiného hlediska, např. složitosti terénu, jak již bylo naznačeno v předchozím rozboru.

2.6 ROBOTOUR 2010

Pátý ročník soutěže autonomních outdoor robotů se bude konat 18. září 2010 na Slovensku, v jednom ze tří předvybraných parků v Bratislavě. Na rozdíl od předešlých ročníků roboty dostanou pouze mapu a souřadnice cíle, nebudou přesně znát svoji startovní polohu a interakce s operátorem se omezí pouze na zadání cíle. Robot úspěšně řešící tuto úlohu by měl být schopen demonstrovat své schopnosti v jakémkoli parku s odpovídající mapou. Zásadní změnou je zadávání požadované trasy, od souborů typu RNDF a MDF organizátoři upustili z důvodů jejich složité

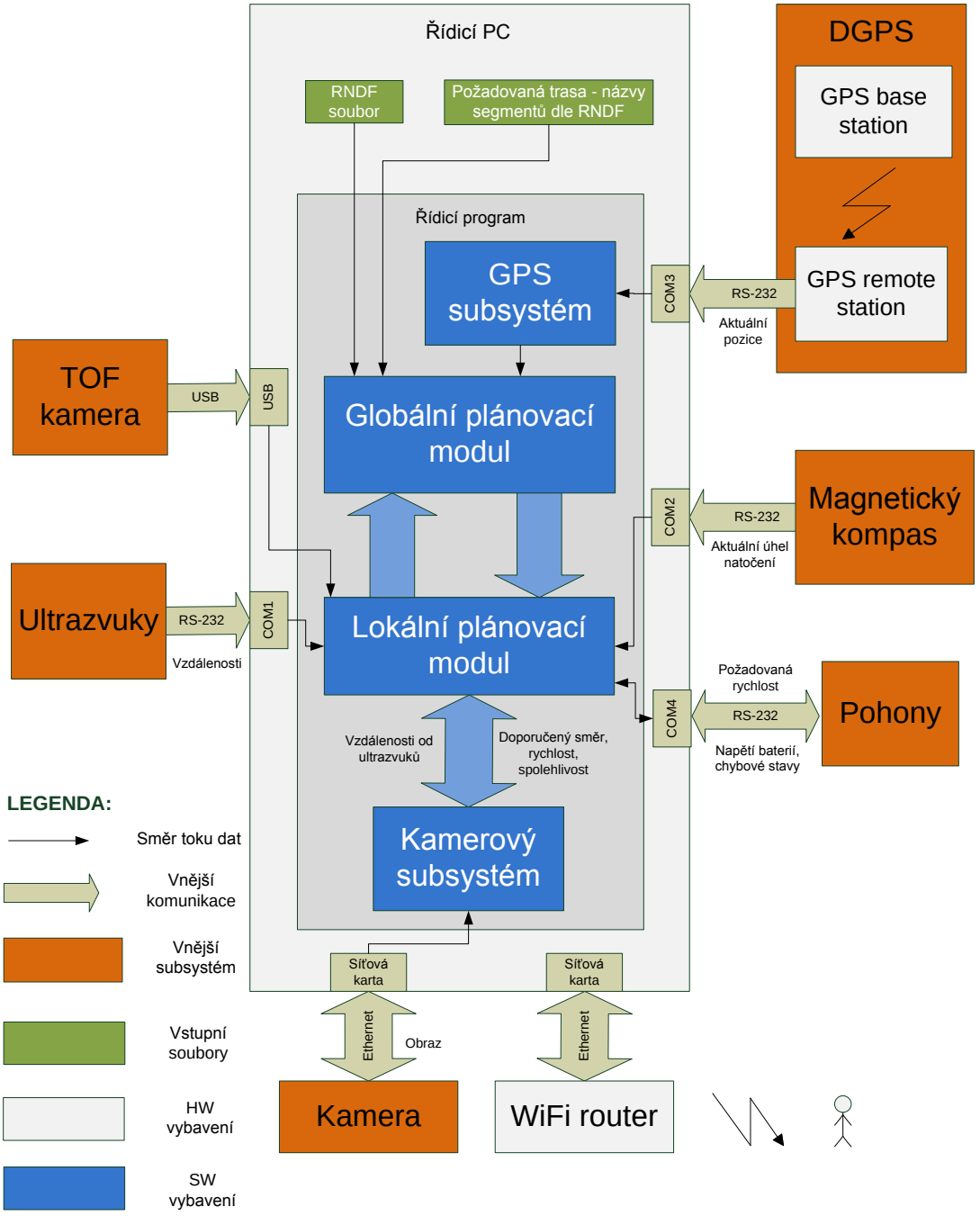
tvorby a byl zaveden koncept zadávání trasy pomocí vektorových map. Důvodem je především volná dostupnost těchto map na internetu. [3]

3. NAVRŽENÝ ŘÍDICÍ SYSTÉM

3.1 ÚVOD

Z teoretického hlediska architektury systému – viz základní pojmy – jde o tzv. hybridní systém, tedy spojení systému reaktivního a mapového, i když mapa je v současné době definována pouze z globálního hlediska. Jde o topologickou mapu reprezentovanou orientovaným grafem, který představuje soubor GPS bodů, jež je třeba pro úspěšné absolvování požadované trasy projíždět v naplánovaném pořadí. Lokalizace je řešena pomocí GPS subsystému, který zajišťuje určení polohy v této mapě zadané RNDF souborem. Plánování probíhá na základě zadané trasy a je rozděleno na dvě samostatné části – lokální a globální plánování.

Blokové schéma navrženého řídicího systému je uvedeno na obrázku 3.1. Je na něm znázorněn základní koncept řídicího systému a také všechny subsystémy a hardwarové prvky zajišťující jeho funkci. Pro komunikaci s jednotlivými subsystémy je z větší části využito rozhraní RS-232 z důvodu jeho jednoduché implementace a dostupnosti.



Obrázek 3.1: Blokové schéma navrženého řídicího systému

3.2 LOKÁLNÍ PLÁNOVÁNÍ

3.2.1 Princip

Lokální plánování zajišťuje plánování pohybu mezi jednotlivými body trasy mobilního robotu. Tyto body jsou naplánovány globálním plánovacím modulem. Pohyb mezi nimi je dle požadavků takový, aby se robot prioritně vyhnul překážkám a nesjel z vyznačené cesty. Informace k tomu potřebné poskytují řídicímu systému především kamerový a ultrazvukový subsystém.

Funkce lokálního plánování je založena na systému priorit a jednoduchých algoritmů navržených již v bakalářské práci [7], ovšem v rámci zlepšení spolehlivosti je kladen větší důraz na spolupráci s kamerovým subsystémem, jehož algoritmy se značně změnily jak po stránce efektivity, tak po stránce spolehlivosti. Nově poskytuje více informací, kromě požadovaného směru i míru spolehlivosti a další.

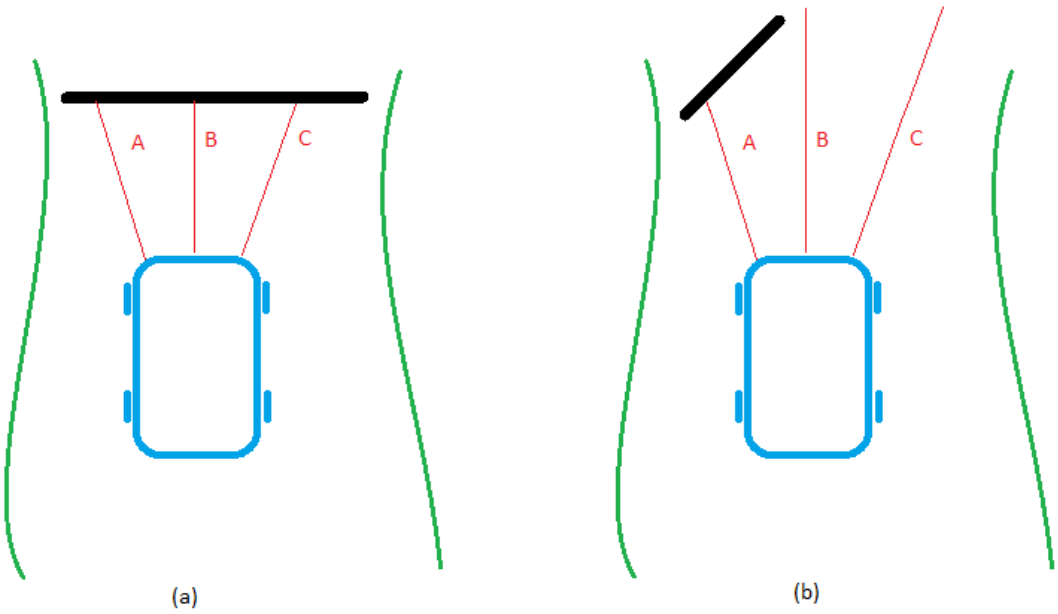
Problémem tohoto principu lokálního plánování je však objíždění překážek založené na modifikovaném „Bug algoritmu“ („VisBug“). Robot tedy objíždí překážku jedním směrem (ve kterém je více volného místa), dokud nemá volnou cestu k dalšímu průjezdnímu bodu. Jelikož mají ultrazvukové senzory vyšší prioritu než pokyny od kamerového subsystému, může se stát, že při objíždění překážky zabírající větší část cesty nezbývá dost místa pro objetí. Robot pak vyjede z cesty, což není přípustné. Problémem je tedy rozpoznání velikosti překážky vzhledem k rozměrům cesty, což však lze rozpoznat při senzorických možnostech daného robotu pouze kamerou. Tam nastávají ovšem další komplikace, např. jak odlišit překážku od stínu atd.

Vývojový diagram lokálního plánování se nachází v příloze 5.

3.2.2 Vyhýbání se překážkám

Dle pravidel soutěže Robotour 2008 bylo třeba rozpoznat překážku přes celou cestu. Na tento problém stávající algoritmus rozpoznávání a objíždění překážek založený na ultrazvukových snímačích nestačil. Bylo tedy třeba navrhnout takovou úpravu, která by byla schopna detekovat překážku přes celou cestu, ovšem s požadavkem nezasahovat do stávajícího rozložení snímačů ani nepřidávat nové. Východisko bylo následující: Pokud robot jede po cestě, resp. po naplánované

trajektorii, neměl by se nikdy dostat do situace, kdy by byly všechny tři ultrazvukové snímače stejnou měrou zastíněny. To se může stát, pouze pokud se robot blíží k rozměrnější překážce, která nebyla předem známa. Takový případ je třeba detekovat. Může se tedy říci, že pokud se vzdálenosti od všech tří předních ultrazvukových snímačů „stejnou měrou“ zmenšují, je velká pravděpodobnost, že se blíží překážka, kterou nebude možno objet. Výrazem „stejnou měrou“ je myšleno, že se absolutní vzdálenosti neliší více než o určitou konstantu. Samozřejmě tento princip nefunguje ve všech případech, ale je to velmi jednoduché řešení postačující pro danou situaci. Tuto situaci ilustruje obrázek 3.2.



Obrázek 3.2: Ilustrace robotu, blížícího se k překážce přes celou cestu (a), k menší překážce (b)

Na základě popsaného teoretického rozboru byla navržena úprava stávajícího algoritmu. Tento algoritmus je tvořen konečným stavovým automatem. Princip spočívá v tom, že byly přidány dva stavy (3. a 4.). První stav detekuje přibližování se k překážce způsobem naznačeným výše. Pokud je takto detekována překážka přes celou cestu, je automat uveden do stavu čtyři, ve kterém provede definovanou činnost – v tomto případě zastavení.

Jelikož však tento způsob detekce není zcela přesný a jednoznačný, byl čtvrtý stav doplněn o časovač, který načasuje přednastavený čas a uvede robot zpět do pohybu, aby se pokusil překážku ještě jednou objet. Vývojový diagram navržené úpravy stávajícího algoritmu je uveden v příloze 3.

3.2.3 Problémy

Jedním z problémů při návrhu algoritmů lokálního plánování je reprezentace a uchování informací ze senzorických subsystémů. Ve stávajícím systému jsou využívána k navigaci pouze aktuálně dostupná data, bylo by však výhodnější uchovávat data ze všech senzorů pro pozdější zpracování či porovnávání. Logicky tento problém vede na vytvoření určité mapy okolního prostředí, z čehož se však již dostáváme do problematiky vytváření map a lokalizace, která je sama o sobě velmi rozsáhlá a částečně byla naznačena v teoretickém úvodu. K vyřešení těchto problémů by bylo možno použít např. mřížek obsazenosti, ovšem to s sebou přináší kromě již zmíněných problémů několik dalších. Jedním z nich je například způsob přesné lokalizace robotu v mřížce. To by bylo možné s využitím dostupných prostředků na daném robotu pouze pomocí GPS, avšak údaje o poloze poskytované tímto systémem nejsou dostatečně přesné pro takovéto účely. Bylo by třeba vyvinout nový odometrický, popřípadě inerciální modul, který by poskytl přesnější data.

3.3 GLOBÁLNÍ PLÁNOVÁNÍ

3.3.1 Úvod

Bylo konstatováno, že lokální plánování řeší průjezd vždy mezi dvěma body. Tyto body je však třeba naplánovat s ohledem na splnění cíle mise, což v tomto případě řeší modul globálního plánování. Zadání požadované trasy je dle pravidel soutěže Robotour řešeno pomocí souborů typu RNDF a MDF, což je standardní formát z americké soutěže DARPA Urban Challenge [10]. Nejdůležitější je soubor RNDF, který obsahuje celou trasu zadanou pomocí GPS souřadnic. Tato trasa je rozdělena na jednotlivé úseky (segmenty), jež mohou tvořit síť různých cest. Pořadí průjezdu určuje podle amerického standardu další soubor – MDF. V našem případě tento soubor zatím nebyl využíván, pořadí průjezdu úseků bylo zadáváno pouze

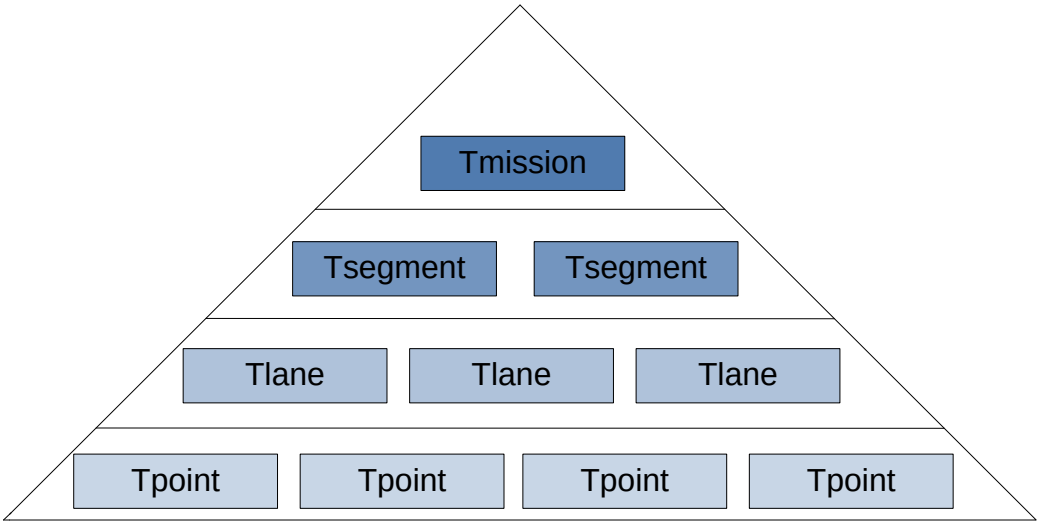
textově jako posloupnost názvů požadovaných segmentů, případně je možno využít navržené automatické plánování. Příklad takové mapy rozdělené na segmenty je možno vidět na obrázku 3.3.

Obrázek 3.3: Příklad mapy rozdělené na segmenty (brněnský park Lužánky)

Pro načítání a uchování dat z již zmíněného RNDF souboru bylo vytvořeno několik tříd v jazyce C++. Obsahují jak struktury pro uchování dat, tak i metody pro práci s nimi. Zároveň usnadňují funkci navrženým plánovacím algoritmům.

3.3.2 Navržené třídy

Dle struktury RNDF souboru byly navrženy celkem čtyři třídy – Tpoint, Tlane, Tsegment a Tmission, které uchovávají a zpracovávají data z RNDF souboru dle požadavků na efektivnost plánovacích algoritmů. Jejich vzájemný vztah zachycuje obrázek 3.4



Obrázek 3.4: Hierarchie navržených tříd pro uchování dat z RNDF souboru

3.3.2.1 Tpoint

Třída Tpoint je základní třídou obsahující všechny potřebné údaje vztažené k jednomu danému bodu, tj. například název daného bodu, jeho pozici, souřadnice v systému UTM atd. Výčet všech proměnných třídy shrnuje tabulka 3.1.

Proměnné třídy		Popis
Typ	Název	
string	sName	Název segmentu
int	iLane	Číslo pruhu
int	iNumber	Pořadové číslo bodu v segmentu
int	iSegment	Číslo příslušného segmentu
double	fNorth	Severní šířka
double	fEast	Východní délka
double	lfNorthing	Souřadnice severní šířky v systému UTM
double	lfEasting	Souřadnice východní délky v systému UTM
Metody		
double pointDistance(Tpoint &point)		Metoda pro výpočet vzdálenosti k zadanému bodu

Tabulka 3.1: Prvky třídy Tpoint

3.3.2.2 Tlane

Třída Tlane podobně jako třída Tpoint obsahuje všechny potřebné údaje vztahující se k jednotlivým pruhům daného segmentu, tj. například název pruhu, příslušnost k segmentu atd. Mimo jiné také obsahuje vektor prvků typu Tpoint definující všechny body obsažené v daném pruhu. Souhrn všech proměnných třídy je uveden v tabulce 3.2.

Proměnné třídy		Popis
Typ	Název	
string	sName	Název pruhu
int	iNumWaypoints	Počet bodů
int	iNumExits	Počet exitů segmentu
int	iLaneWidth	Šířka pruhu
int	iSegment	Číslu příslušného segmentu
double	fLength	Délka pruhu
vector<Tpoint>	vtpExit	Seznam možných exitů z pruhu
vector<Tpoint>	vtpWaypoints	Vektor všech bodů
Tpoint	tpStartPoint	Počáteční bod
Tpoint	tpEndPoint	Koncový bod

Tabulka 3.2: Prvky třídy Tlane

3.3.2.3 Tsegment

Třída Tsegment opět obsahuje informace vztahující se k jednotlivým segmentům, viz tabulka 3.3. Hlavním prvkem je vektor pruhů, které náleží k danému segmentu. Ve většině případů jsou tyto pruhy dva, pokud je daný segment průjezdný pouze jednosměrně, může obsahovat pouze jeden pruh.

Proměnné třídy		Popis
Typ	Název	
string	sName	Název segmentu
int	iNumLanes	Počet pruhů
int	iNumber	Číslo segmentu
vector<Tlane>	vtlLanes	Vektor pruhů

Tabulka 3.3: Prvky třídy Tsegment

3.3.2.4 Tmission

Třída Tmission obsahuje informace o naplánované trase, jejich souhrn se nachází v tabulce 3.4. Trasa je naplánována plánovacími algoritmy, které jsou popsány dále. Pro následnou navigaci v této trase je nejdůležitější částí vektor pruhů, které musí robot projet pro dosažení požadovaného cíle.

Proměnné třídy		Popis
Typ	Název	
double	lfLength	Délka mise
int	iNumLanes	Počet pruhů
int	iNumPoints	Celkový počet bodů
string	sName	Název mise
vector<double>	vdSegmentMatrix	Matice sousednosti
vector<Tlane>	vtlLanes	Dynamické pole pruhů

Tabulka 3.4: Prvky třídy Tmission

Všechny výše uvedené třídy mají také definovány základní konstruktory a operátory potřebné pro jejich korektní funkčnost.

3.3.3 Načítání dat z RNDF souboru

Pro načítání dat z RNDF souboru slouží funkce *decodeRNDFtoWay*, jejíž prototyp je ve tvaru:

```
vector<Tsegment> decodeRNDFtoWay(string nameRNDF);
```

Tato funkce bere jako parametr cestu k požadovanému RNDF souboru a vrací vektor prvků typu Tsegment, který obsahuje všechny segmenty definované v RNDF souboru. Její úkol v podstatě spočívá pouze v dekódování dat obsažených v RNDF souboru do datových tříd Tpoint, Tlane a Tsegment.

3.3.4 Plánovací algoritmus

Pro naplánování trasy z vektoru segmentů typu Tsegment slouží metoda s názvem *missionPlan*. Její prototyp má tvar:

```
int missionPlan(string sPath, Tmission *TmMission, vector<Tsegment>
*tsSegment);
```

Metoda přijímá tyto parametry:

- Textový řetězec s názvy požadovaných segmentů (např. A, B, C“).

- Ukazatel na prvek typu Tmission, zahrnující posloupnost bodů, které je třeba projet pro úspěšné splnění mise.
- Prvek typu Tsegment, obsahující dekódované segmenty z RNDF souboru.
- Návratová hodnota typu *int*, definující případnou chybu ve standardním formátu – 0 odpovídá bezchybnému zpracování, další kladné hodnoty vyjadřují, jaký typ chyby nastal:
  - 0 – Žádná chyba.
  - 1 – Špatně zadaná trasa.
  - 2 – Nenalezeny požadované segmenty v daném RNDF.
  - 3 – Nenalezena požadovaná trasa v daném RNDF.

Plánování probíhá v podstatě jako prohledávání stavového prostoru, který je tvořen jednotlivými segmenty typu Tsegment. Tento datový typ obsahuje mimo jiné i návaznost na další segmenty v podobě tzv. exitů (výjezdů). Výjezdy se skládají z názvů bodů ostatních segmentů a bodů daného segmentu, které na sebe navazují. Prohledávání je poté velmi jednoduché, ztěžuje ho ovšem přítomnost pruhů. Ty jsou v původním formátu zavedeny kvůli „klasickým“ silnicím, které mají více pruhů. V našem případě parkových cestiček jsou tyto pruhy využívány pouze k definování požadovaného směru průjezdu jednotlivými segmenty.

3.3.5 Pomocné metody

Navržená třída dále obsahuje několik pomocných metod pro usnadnění práce s daty.

První z nich je metoda *convCoor* :

```
void convCoor(string Sbod1, double *fNorth, double *fEast);
```

Tato metoda převádí vstupní řetězec ve tvaru "50°6'15.007N, 14°25'36.838E" na dvě reálná čísla.

Další je metoda *dist*, která počítá vzdálenosti dvou prvků třídy Tpoint:

```
double dist(Tpoint a, Tpoint b)
```

3.3.6 Načítání dat z vektorových map

Jak již bylo naznačeno v úvodu, pro soutěž Robotour 2010 nastává jedna zásadní změna v zadávání trasy. Organizátoři opustili koncept RNDF souborů, pro jejich složitou tvorbu. Trasa bude zadávána pomocí vektorových map dostupných na internetu. Z toho vyplývá požadavek dekódování trasy z těchto vektorových map. Specifikaci použitého formátu OpenStreetMap lze nalézt například v [6].

Pro požadavky stávajícího plánování bylo třeba tyto vektorové mapy upravit. Protože je plánovací algoritmus navržen na zpracování souborů typu RNDF, byl kolegou zpracován program k převodu těchto vektorových map na RNDF formát, což podstatně zjednodušilo potřebnou úpravu plánování.

V plánovacím algoritmu však i přesto bylo třeba udělat několik úprav – především ve způsobu načítání dat. Plánování průjezdu jednotlivých segmentů bylo možno zachovat.

3.4 AUTOMATICKÉ PLÁNOVÁNÍ PRŮJEZDU SEGMENTŮ TRASY

3.4.1 Princip

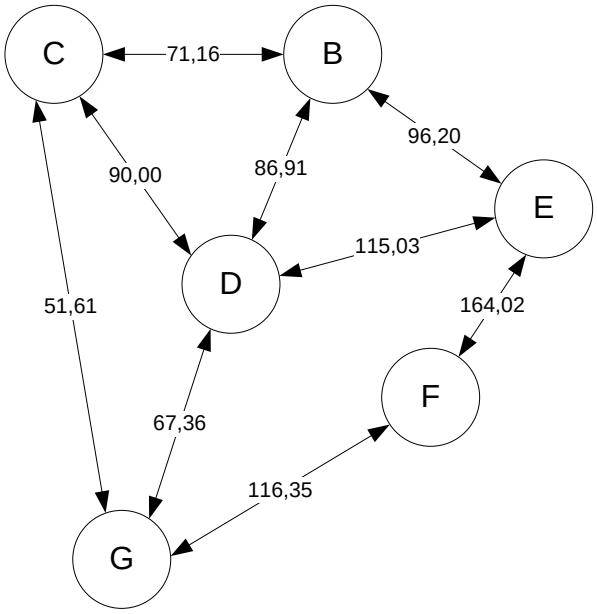
Automatickým plánováním je v této části myšleno plánování průjezdu jednotlivých segmentů trasy řídicím systémem autonomně za pomoci grafových algoritmů uvedených v kapitole 2.5. Plánovaná trasa může být optimální z hlediska zadané kriteriální funkce, např. nejkratší trasy, nejpravděpodobnějšího správného projetí jednotlivých segmentů, nejmenší energetické náročnosti atd. Toto kritérium musí být zadáno před samotným plánováním. Navržený algoritmus předpokládá znalost polohy robotu, zmíněné kriteriální funkce a cílového bodu/segmentu.

Z teoretického hlediska se jedná o plánování optimální trasy, dle kritérií zmíněných výše, v ohodnoceném orientovaném grafu. Dané kritérium tvoří ohodnocení grafu, jehož vrcholy jsou tvořeny jednotlivými segmenty.

Jako kritérium byla v tomto případě zvolena délka segmentů, samozřejmě je možno zvolit jiné i různá kritéria kombinovat. Pro jednoduchou implementaci je vzdálenost dvou vedlejších segmentů definována jako součet polovin jejich délek.

Na obrázku 3.5 je vidět příklad orientovaného grafu pro část mapy z obrázku 3.3. Písmena v jednotlivých vrcholech značí dané segmenty, číslice na hranách vyjadřují vzdálenosti udané v metrech. K tomuto se váže tabulka 3.5, která prakticky ukazuje matici sousednosti využitou v grafovém algoritmu.

Pro vyhledání nejkratší trasy v daném ohodnoceném grafu byl dle teoretického úvodu zvolen Dijkstrův algoritmus. Jeho implementace je popsána v následující kapitole.



Obrázek 3.5: Příklad orientovaného grafu pro část mapy z Obrázek 3.3

vzdálenosti [m]	B	C	D	E	F	G
B	-	71,16	86,91	96,20	-	-
C	71,16	-	90,00	-	-	51,61
D	86,91	90,00	-	115,03	-	-
E	96,20	-	115,03	-	164,02	-
F	-	-	-	164,02	-	116,35
G	-	51,61	67,36	-	116,35	-

Tabulka 3.5: Příklad tabulky vzdáleností uzlů orientovaného grafu

3.4.2 Softwarové řešení

Programové řešení tohoto plánování zajišťuje funkce *dynamicMissionPlan*, jejíž prototyp má tvar:

```
int dynamicMissionPlan(string sTarget, Tmission *tmMission,
vector<Tsegment> &segment, Tpoint tpClosestPoint);.
```

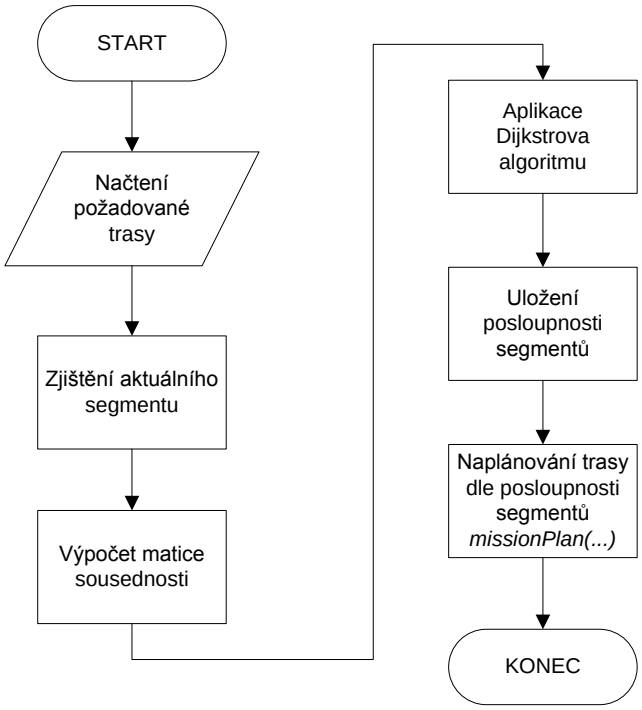
Parametry této funkce jsou:

- Textový řetězec *sTarget*, který může obsahovat buď pouze název cílového segmentu uvozený znakem „-“, nebo i název startovního segmentu. Např. „-G,A“ – požadavek na naplánování trasy ze segmentu A do G. Pokud není startovací segment zadán, je automaticky zvolen aktuální segment dle souřadnic z GPS.
- Ukazatel na proměnnou *tmMission*, do níž bude uložena naplánovaná mise.
- Vektor všech segmentů *segment*.
- Aktuální pozice z GPS ve struktuře Tpoint pro zjištění aktuálního segmentu *tpClosestPoint*.
- Návratová hodnota typu *int* obsahuje případnou chybu stejného typu jako funkce *missionPlan*.

Navržená funkce se skládá ze tří částí. V první je vypočtena matice sousednosti z předzpracovaného RNDF souboru, obsaženého v proměnné *segment*. Dále je pomocí informací od GPS zjištěn aktuální segment, ve kterém se robot nachází. Pokud není signál k dispozici, či je požadováno naplánovat trasu z určitého segmentu, je možno zadat jak cílový, tak i výchozí segment. V případě nedostupnosti ani jedné z předcházejících informací je jako startovní zvolen první segment. Název cílového segmentu je třeba zadat ve vstupním řetězci jako první a uvodit jej znakem „-“. Startovní segment následuje a je oddělen čárkou.

Druhá část zpracovává samotný Dijkstrův algoritmus [13], výstupem je posloupnost segmentů, potřebných k dosažení požadovaného cíle nejkratší možnou cestou.

Tato posloupnost je zpracována ve třetí části již zmiňovanou funkcí *missionPlan*, která naplní proměnnou tmMission posloupností bodů (pruhů), odpovídajících splnění daného cíle. Vývojový diagram této funkce je uveden na obrázku 3.6.



Obrázek 3.6: Vývojový diagram automatického plánování

3.5 NAVIGACE

Princip navigace na základě předzpracovaného RNDF souboru je po předzpracování plánovacími algoritmy již velice jednoduchý. Při zahájení navigace je dle GPS, pokud je dostatečně silný signál, nalezen nejbližší bod naplánované trasy, a ten je považován za startovní. Následuje výpočet úhlu k dalšímu bodu a případně načtení následujícího bodu při dosažení určité vzdálenosti od požadovaného bodu. Tato data jsou předána pomocí „tabule“ lokálnímu plánovacímu algoritmu, který rozhodne o aktuální trase. Vývojový diagram je znázorněn v příloze 4.

3.6 SHRNUTÍ

V této kapitole byly popsány samotné plánovací algoritmy. Byly rozděleny na lokální a globální. Globální plánování je možné buď pomocí přesně zadaného pořadí segmentů, nebo prostřednictvím automatického plánování z počátečního segmentu do koncového na základě Dijkstrova algoritmu. Algoritmy vycházejí z mapy definované RNDF souborem, nový směr zadávání map v soutěži Robotour je však směrem k vektorovým mapám. To bylo vyřešeno algoritmem pro převod zadaných vektorových map na formát RNDF, kterým se zabýval kolega.

4. ŘÍDICÍ APLIKACE

4.1 ÚVOD

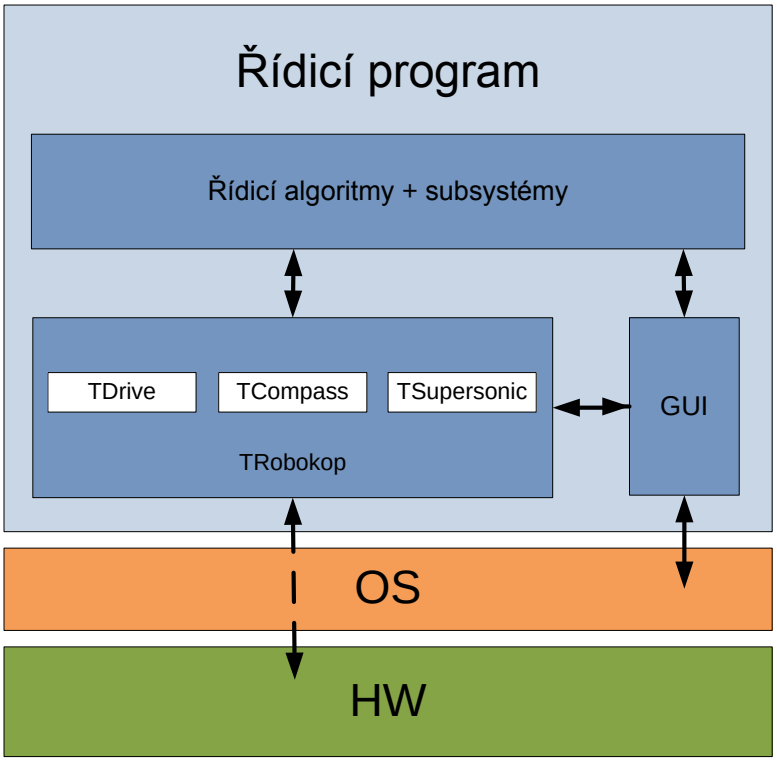
Řídicí program je postaven na základech navržených již v předchozí bakalářské práci [7]. Skládá se z grafického rozhraní a samotných řídicích algoritmů. Jeho princip je uveden na obrázku 3.1. Dále obsahuje třídy pro komunikaci se subsystémy na nižší úrovni. Dále je také propojen s řídicími programy některých dalších subsystémů na vyšší úrovni, a to kamerového a GPS. Hierarchie jednotlivých zdrojových souborů je uvedena v příloze 2.

V této kapitole jsou popsány jednak zmíněné třídy pro komunikaci s jednotlivými subsystémy, jednak grafické rozhraní aplikace a její ovládání.

4.2 TŘÍDY PRO OVLÁDÁNÍ ROBOTU

Veškeré řízení subsystémů robotu bylo implementováno do několika tříd, které zajišťují jednoduchou obsluhu a komunikaci s těmito subsystémy. Komunikace se samotným hardwarem je zajišťována prostřednictvím operačního systému řídicího počítače.

Předávání dat mezi jednotlivými subsystémy probíhá pomocí sdílených proměnných. V literatuře [9] se tento způsob nazývá tabule. V postatě jde o to, že informace jsou různými subsystémy ukládány na jedno místo – tabuli (v našem případě globální proměnné), odkud si je mohou ostatní subsystémy kdykoliv zjistit.



Obrázek 4.1: Architektura datových tříd pro řízení robotu

Obrázek 4.1 znázorňuje postavení navržených tříd pro řízení robotu vzhledem k řídicímu programu, HW robotu a OS. Detailnější struktura řídicího programu je na obrázku 3.1.

Všechny navržené třídy implementují základní metody zajišťující funkčnost daných tříd a komunikaci se subsystémy. Mezi ně patří zejména konstruktory a destruktor daných tříd a dále metody *Run* a *Stop*, které zahajují a ukončují komunikaci se subsystémy. Pro připojení požadovaného subsystému vždy slouží metoda *Run*. Spustí obslužné vlákno pro komunikaci a otevře příslušný komunikační port a poté ve vlákně spustí nekonečnou smyčku obsluhující daný subsystém. Pro ukončení komunikace je implementována metoda *Stop*, která přeruší běh obslužného vlákna a zajistí korektní ukončení komunikace.

4.2.1 Třída TCompasCmps03

Tato třída zajišťuje potřebné rozhraní se subsystémem elektronického kompasu typu CMPS03. Soupis proměnných a metod, které třída obsahuje, je uveden v tabulce 4.1.

Proměnné třídy		Popis
Typ	Název	
int	_angle	Aktuální úhel natočení
int	_x	Zrychlení v ose x
int	_y	Zrychlení v ose y
int	_z	Zrychlení v ose z
int	compassCorr	Korekce hodnoty úhlu natočení
HANDLE	_hThread	Pomocná proměnná pro běh vlákna
bool	_stop	Příznak pro ukončení vlákna
int	_baudRate	Proměnná pro práci se sériovým portem
string	_comName	Proměnná pro práci se sériovým portem
bool	_connected	Příznak korektního připojení subsystému
Tserial *	_compassComPort	Proměnná pro práci se sériovým portem
Metody		
int Run()		Metoda pro spuštění obsluhy subsystému
int Stop(DWORD timeout)		Metoda pro ukončení obsluhy subsystému
bool IsRunning()		Metoda pro zjištění běhu
bool IsConnected()		Metoda pro ověření připojení subsystému
int GetAngle()		Zjištění aktuálního úhlu natočení

Tabulka 4.1: Proměnné třídy TCompassCMPS03

4.2.2 Třída TDrive

Třída TDrive je určena pro komunikaci s pohonným subsystémem. Jednak zajišťuje ovládání tohoto subsystému, jednak také přijímá informace o stavu robotu, jež tento subsystém poskytuje. Jedná se především o napětí baterií a stav pohonů. Proměnné a metody třídy jsou shrnuty v tabulce 4.2.

Proměnné třídy		Popis
Typ	Název	
Int	_iSpeedA	Rychlost motoru A
Int	_iSpeedB	Rychlost motoru B
Int	_voltage	Korekce hodnoty úhlu natočení
Char	_cSignA	Směr otáčení motoru A
Char	_ cSignB	Směr otáčení motoru B
HANDLE	_hThread	Pomocná proměnná pro běh vlákna
Bool	_stop	Příznak pro ukončení vlákna
int	_baudRate	Proměnná pro práci se sériovým portem
String	_comName	Proměnná pro práci se sériovým portem
Bool	_connected	Příznak korektního připojení subsystému
Tserial *	_ driveComPort	Proměnná pro práci se sériovým portem
Metody		
int Run()		Metoda pro spuštění obsluhy subsystému
int Stop(DWORD timeout)		Metoda pro ukončení obsluhy subsystému
bool IsRunning()		Metoda pro zjištění běhu
bool IsConnected()		Metoda pro ověření připojení subsystému
bool IsInitialized()		Metoda pro ověření inicializace subsystému
bool IsOvercurrent()		Metoda pro ověření nadproudu motoru
int GetSpeedA ()		Zjištění aktuální rychlosti motoru A
int GetSpeedB ()		Zjištění aktuální rychlosti motoru B
int GetVoltage ()		Zjištění aktuálního napětí baterií
int SetSpeedA (int speed, char sign)		Nastavení aktuální rychlosti motoru A
int SetSpeedB (int speed, char sign)		Nastavení aktuální rychlosti motoru B
int Init ()		Metoda spouštěná po připojení napájení
int OverCurrent()		Metoda obsluhující stav nadproudu

Tabulka 4.2: Proměnné třídy TDrive

4.2.3 Třída TSupersonic

Třída Tsupersonic implementuje rozhraní se subsystémem ultrazvukových snímačů, od kterých především přijímá naměřené vzdálenosti. Proměnné třídy a metody zajišťující její funkčnost, viz tabulka 4.3.

Proměnné třídy		Popis
Typ	Název	
Int	_A	Rychlost motoru A
Int	_B	Rychlost motoru B
Int	_C	Korekce hodnoty úhlu natočení
Char	_Z	Směr otáčení motoru A
Char	_baudRate	Směr otáčení motoru B
HANDLE	_LENGTH	Pomocná proměnná pro běh vlákna
Bool	_stop	Příznak pro ukončení vlákna
Bool	_connected	Příznak korektního připojení subsystému
Tserial *	_ supersonicComPort	Proměnná pro práci se sériovým portem
Metody		
int Run()		Metoda pro spuštění obsluhy subsystému
int Stop(DWORD timeout)		Metoda pro ukončení obsluhy subsystému
bool IsRunning()		Metoda pro zjištění běhu
bool IsConnected()		Metoda pro ověření připojení subsystému
int GetA ()		Zjištění aktuální vzdálenosti ultrazvuku A
int GetB ()		Zjištění aktuální vzdálenosti ultrazvuku B
int GetC ()		Zjištění aktuální vzdálenosti ultrazvuku C
int GetZ ()		Zjištění aktuální vzdálenosti ultrazvuku Z

Tabulka 4.3: Proměnné třídy TSupersonic

4.2.4 GPS

Pro uchování dat z GPS subsystému slouží struktura GPSSTRUCT, obsahující všechny informace potřebné pro GPS navigaci. Tato struktura byla společně s funkcemi pro obsluhu GPS subsystému navržena kolegou, který se zabývá zpracováním signálů od senzorů robotu na nižší úrovni. Řídicí systém však tuto strukturu a funkce implementuje a využívá je pro komunikaci s GPS subsystémem a pro plánování.

4.3 RUČNÍ OVLÁDÁNÍ

Pro pohodlnější ovládání robotu bez nutnosti připojovat periferie, či notebook pomocí Wi-Fi bylo do řídicí aplikace implementováno řízení pomocí bezdrátového gamepadu. Programové zpracování jeho povelů je založeno na rozhraní DirectX operačního systému, které zajišťuje přenos povelů z gamepadu do řídicí aplikace.

Obrázek 4.2: Bezdrátový gamepad Logitech Rumblepad 2 [5]

Implementovány byly dva druhy řízení – pomocí jednoho analogového ovladače a pomocí dvou analogových ovladačů, přičemž se ovládají buď oba motory zároveň, nebo každý zvlášť. Jednotlivým tlačítkům byly přiřazeny různé funkce ovládající chování robota a jeho základní nastavení. Souhrn těchto funkcí a jim příslušejících tlačítek je uveden v tabulce 4.4.

Tlačítko	Funkce
1	Režim ručního řízení 1
2	Režim ručního řízení 2
1+2	Zapnutí/vypnutí ochrany baterií
3	Start/stop učení kamerového subsystému
4	Start/stop kamerové navigace
5	Uložení aktuálního bodu
6	Start navigace přes ručně uložené body
7+8	Automatický režim
6+7+9+10	Vypnutí robotu
5+6+9+10	Restart robotu
9	Aktivace/deaktivace kamery v automatickém režimu
10	Aktivace/deaktivace GPS v automatickém režimu

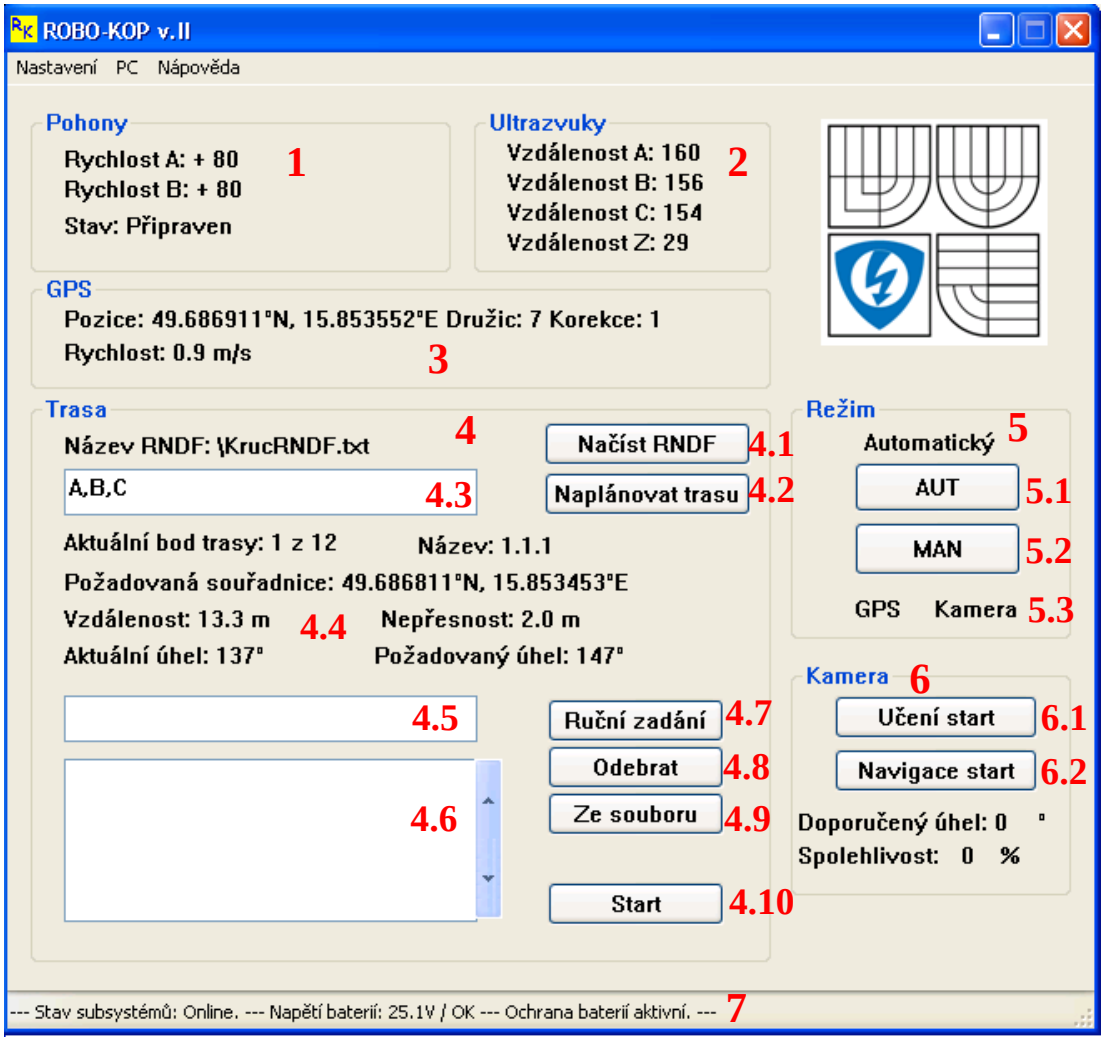
Tabulka 4.4: Seznam funkcí přiřazených k jednotlivým tlačítkům gamepadu

4.4 GUI

Grafické rozhraní bylo přepracováno s ohledem na jednodušší a efektivnější programování. Je však opět velmi prosté a přehledné, orientované především na usnadnění ladění a testování řídicích algoritmů. Zobrazuje potřebná data ze senzorických subsystémů a umožňuje některá nastavení systému robotu. Lze ho ovšem využít i jako operátorské stanoviště, neboť umožňuje například ukládání aktuálních GPS souřadnic a jejich opětovné autonomní projetí. Dále je možno načíst data z daného RNDF souboru a zadat požadovanou trasu, která je naplánována pomocí funkcí navržených v kapitole 3.2. Grafické rozhraní bylo také rozšířeno o okno s možností nastavování různých parametrů robotu, jako např. maximální rychlost, reakční vzdálenosti ultrazvuků atd.

4.4.1 Hlavní okno

Hlavní okno aplikace zobrazuje základní informace dostupné od senzorických subsystémů. Vzhled okna je vidět na obrázku 4.3. Dále okno také obsahuje prvky, pomocí kterých je možno ovládat různé režimy robotu dle popisu uvedeného dále.



Obrázek 4.3: Okno uživatelského rozhraní řídicího systému

Popis jednotlivých částí:

1 - Zobrazování informací o pohonném subsystému, tj. o rychlosti jednotlivých motorů a celkovém stavu subsystému. Stav může nabývat hodnot připojeno, odpojeno, vypnuto ovládací napětí (CS) a připraven.

2 - Vzdálenosti měřené pomocí ultrazvukových snímačů – tři vpředu (při pohledu shora – zleva A, B, C) a jeden vzadu (Z).

3 - Informace od GPS subsystému, poloha, počet satelitů, stav atd.

4 - Informace o a prvky umožňující plánování.

4.1 - Tlačítko sloužící pro výběr a načtení RNDF souboru.

4.2 - Tlačítko k naplánování požadované trasy.

4.3 - Pole pro zadávání požadované trasy, buď jako posloupnost segmentů, nebo pouze cílový segment v případě automatického plánování (viz kapitola 3.4.2).

4.4 - Informace o aktuálním/následujícím bodu, vzdálenost mezi nimi a aktuální/požadovaný úhel.

4.5 - Pole pro ruční zadávání požadovaného bodu trasy.

4.6 - Tabulka zobrazující posloupnost bodů ručně zadávané trasy.

4.7 - Tlačítko pro uložení ručně zadaného bodu. Pokud nejsou souřadnice bodu zadány v poli 4.5, je uložena aktuální pozice robotu.

4.8 - Tlačítko pro odebrání ručně zadaného bodu.

4.9 - Tlačítko pro načtení posloupnosti bodů ze souboru.

4.10 - Start navigace přes ručně zadané body.

5 - Část volby režimů.

5.1 - Tlačítko volby automatického režimu.

5.2 - Tlačítko volby manuálního režimu.

5.3 - Zobrazování subsystémů podílejících se na navigaci v automatickém režimu. Jejich volba se provádí v okně nastavení nebo pomocí klávesových zkratek.

6 - Část zobrazující informace o kamerovém subsystému.

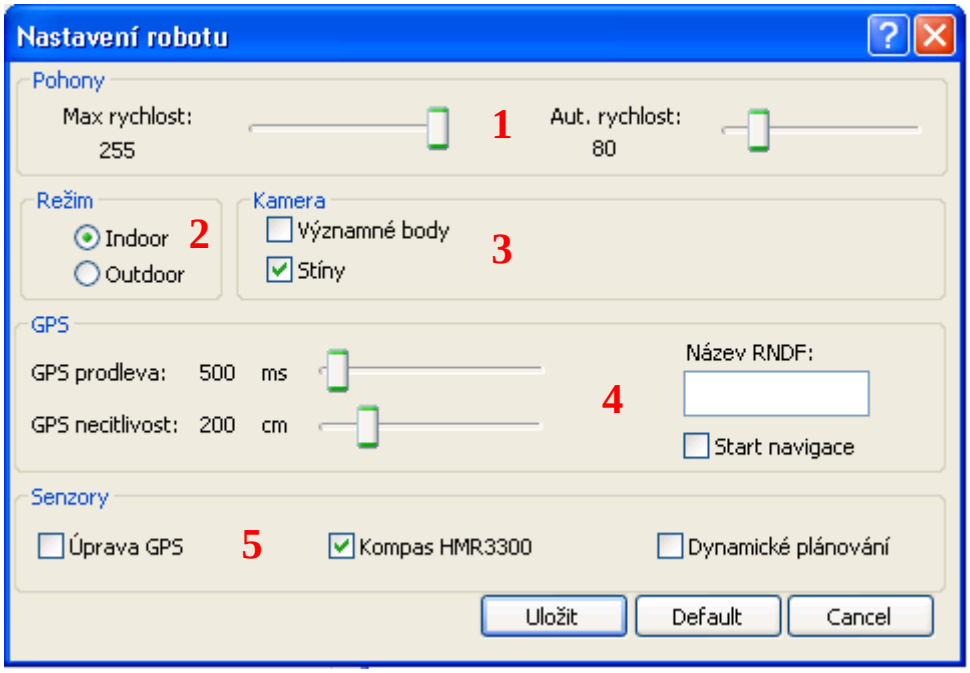
6.1 - Tlačítko pro zahájení učicí fáze kamerového subsystému v režimu významných bodů, aktivace režimu se provádí v okně nastavení.

6.2 - Tlačítko pro spuštění kamerové navigace v obou režimech.

7 - Stavový řádek, zobrazující informace o celkovém stavu subsystémů a napětí baterií robotu.

4.4.2 Okno nastavení

K nastavení základních parametrů pro provoz a testování robotu slouží okno nastavení (viz obrázek 4.4). Toto okno je rozděleno na části podobně jako hlavní okno aplikace.



Obrázek 4.4: Okno nastavení parametrů

Popis jednotlivých částí:

1 - Nastavení maximální rychlosti robotu (0-255).

2 - Režim jízdy – pro vnitřní a venkovní provoz. Tím je ovlivněna především snímací vzdálenost ultrazvukových snímačů v automatickém režimu a také rychlosti pohonů.

3 - Nastavení parametrů kamerového subsystému.

4 - Parametry GPS subsystému – prodleva mezi aktualizacemi polohy a nepřesnost při projíždění zadaných GPS bodů.

5 - Nastavení některých parametrů ostatních senzorů.

Kromě grafického rozhraní samotné řídicí aplikace lze také pomocí tlačítek 6.1 a 6.2 (viz obrázek 4.3), zobrazit grafické rozhraní kamerového subsystému, zachycené na obrázku 4.5. Pomocí přepínače v okně nastavení lze volit mezi režimy

kamerového subsystému, a to segmentace obrazu a využití významných bodů v obraze. Podrobnosti lze nalézt v práci kolegy, zabývajícím se kamerovým subsystémem.

Obrázek 4.5: Okno zpracování obrazu kamery - OpenCV

4.5 KLÁVESOVÉ ZKRATKY

Pro jednodušší ovládání robotu byly již v předchozí práci implementovány klávesové zkratky pro nejdůležitější funkce. Ty byly v této práci rozšířeny o další funkce a jejich soupis je uveden v tabulce 4.5. Stisknutí kláves robot signalizuje zvukovými signály.

Klávesa	Funkce
Šipka nahoru	Ovládání směru jízdy
Šipka dolů	Ovládání směru jízdy
Šipka doprava	Ovládání směru jízdy
Šipka doleva	Ovládání směru jízdy
Mezerník	Zastavení
a	Automatický režim
m	Manuální režim
g	Aktivace/deaktivace gps v automatickém režimu
k	Aktivace/deaktivace kamery v automatickém režimu
n	Aktivace/deaktivace vypnutí robotu při nízkém napětí
s	Start/stop kamerové navigace
u	Start/stop učení kamerového subsystému
Ctrl+Alt+Enter	Ukončení systému Windows a vypnutí

Tabulka 4.5: Klávesové zkratky pro ovládání robotu

4.6 SHRNUTÍ

Řídicí program je napsán v programovacím jazyce C++ pomocí programovacího prostředí Microsoft Visual C++ 2008 a běží na operačním systému Microsoft Windows XP. Samotný řídicí program obsahuje přibližně 3500 řádků zdrojového kódu a je rozdělen na jednotlivé zdrojové soubory, jak je uvedeno v příloze 2.

Grafické rozhraní obsahuje dvě základní okna – hlavní okno a okno nastavení. Hlavní okno zobrazuje informace poskytované subsystémy a umožňuje ovládání robotu. V okně nastavení je možno upravovat volitelné parametry řídicího systému, ale i některých subsystémů robotu.

Ovládání robotu je možné v zásadě třemi způsoby. Přímo přes grafické rozhraní, pomocí klávesnice a klávesových zkratek, nebo pomocí bezdrátového ovladače a tlačítek na něm umístěných.

5. ZÁVĚR

V této diplomové práci bylo v návaznosti na předchozí bakalářskou práci pokračováno ve vývoji řídicího systému. Zpracovány byly především plánovací algoritmy a třídy zajišťující komunikaci se subsystémy mobilního robotu typu Minidarpa s pracovním názvem RoboKop. Tento robot je výsledkem kolektivní práce celého týmu a po navržených úpravách řídicího systému by měl být schopen zúčastnit se soutěže Robotour 2010. Jeho současný stav je vidět na obrázku 5.1, kde je zachycen při soutěži Robotour 2009, ve které se umístil na 4. místě a získal cenu diváků.

Navržený plánovací algoritmus byl rozdělen na dvě hlavní části – lokální a globální. V lokálním plánování byly upraveny především algoritmy pro spolupráci s kamerovým subsystémem a také algoritmy pro objíždění překážek pomocí ultrazvukových snímačů. Pozornost byla však soustředěna především na globální plánování, které bylo třeba dle požadavků daných soutěží Robotour přednostně vyřešit. Byly navrženy a implementovány funkce pro plánování průjezdních bodů jak podle pevně zadané požadované trasy, tak i pomocí automatického plánovacího algoritmu. Tomuto algoritmu stačí zadat požadovaný cíl v daném RNDF souboru. Následně je vhodná trasa (s ohledem na zvolené kritérium) naplánována pomocí Dijkstrova algoritmu. Tyto plánovací algoritmy jsou popsány ve třetí kapitole.

Celý návrh řídicího systému byl směřován především k větší modularitě pro lepší podmínky dalšího vývoje. Z toho důvodu byly vytvořeny třídy v jazyce C++ sloužící pro ovládání a komunikaci se subsystémy robotu. Tyto třídy jsou společně s grafickým rozhraním a dálkovým ovládáním robotu popsány v poslední kapitole.

Vývoj řídicího systému jako celku však není zdaleka ukončen. V mnoha oblastech je stále co dodělávat a zlepšovat. V případě dalšího vývoje by bylo vhodné se zaměřit na zlepšení metod lokálního plánování, které jsou dle mého názoru v současnosti slabým článkem celého systému.

Obrázek 5.1: Robot RoboKop na soutěži Robotour 2009

6. SEZNAM POUŽITÉ LITERATURY

[1] ALBUS, James S., et al. *NASREM THE NASA/NBS STANDARD REFERENCE MODEL FOR TELEROBOT CONTROL SYSTEM ARCHITECTURE*. [online]. 1994, [cit. 2010-03-10]. Dostupný z WWW: http://www.isd.mel.nist.gov/documents/albus/Loc_143.pdf

[2] ČERNÝ, Jakub. *Základní grafové algoritmy* [online]. [s.l.], 2008. 168 s. Skriptum. MFF UK. Dostupné z WWW: http://kam.mff.cuni.cz/~kuba/ka.

[3] DLOUHÝ, Martin, WINKLER, Zbyněk. *Robotour 2010* [online]. 2009-11-23 [cit. 2009-12-08]. Dostupný z WWW: http://robotika.cz/competitions/robotour2010/cs.

[4] HOLLAND, John. *Designing Autonomous Mobile Robots*. [s.l.] : Newnes, c2004. 335 s. ISBN 0-7506-7683-3.

[5] Internetové stránky firmy Logitech. http://www.logitech.com

[6] *JOSM file format* [online]. 2007-2009 , 14. 11. 2009 [cit. 2010-01-07]. Dostupný z WWW: http://wiki.openstreetmap.org/wiki/JOSM_file_format.

[7] KOPECKÝ, Martin. *Programové vybavení mobilního robotu Minidarpa.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 50 s. Vedoucí bakalářské práce Ing. Lukáš Kopečný.

[8] SIEGWART, Roland, NOURBAKHSH, Illah R. *Introduction to Autonomous Mobile Robots*. Cambridge(Mass.) : The MIT Press, c2004. 321 s. ISBN 0-262-19502-X.

[9] ŠTĚPÁN, Petr. *Vnitřní reprezentace prostředí pro autonomní mobilní roboty* [online]. [s.l.], 2001. 114 s. Dizertační práce. ČVUT.

[10] *Urban Challenge : Route Network Definition File (RNDF) and Mission Data File (MDF) Formats* [online]. March 14, 2007 [cit. 2009-04-19]. Dostupný z WWW: http://www.darpa.mil/GRANDCHALLENGE/docs/RNDF_MDF_Formats_031407.pdf.

[11] Wikipedia. *DARPA Grand Challenge* [online]. 2003-2009, 30. 12. 2009 [cit. 2010-01-07]. Dostupný z WWW: http://en.wikipedia.org/wiki/DARPA_Grand_Challenge.

[12] WINKLER, Zbyněk. *Robotika.cz* [online]. 2003 [cit. 2010-02-15]. Plánování na mřížce. Dostupné z WWW: http://robotika.cz/guide/gridplan/cs .

[13] *World of Brock* [online]. 2009 [cit. 2010-04-15]. How to implement Dijkstra's algorithm. Dostupné z WWW: http://worldofbrock.blogspot.com/2009/10/how-to-implement-dijkstras-algorithm.html .

7. SEZNAM PŘÍLOH

Příloha 1: Část ukázkového RNDF souboru parku Lužánky – viz Obrázek 3.3. [3]

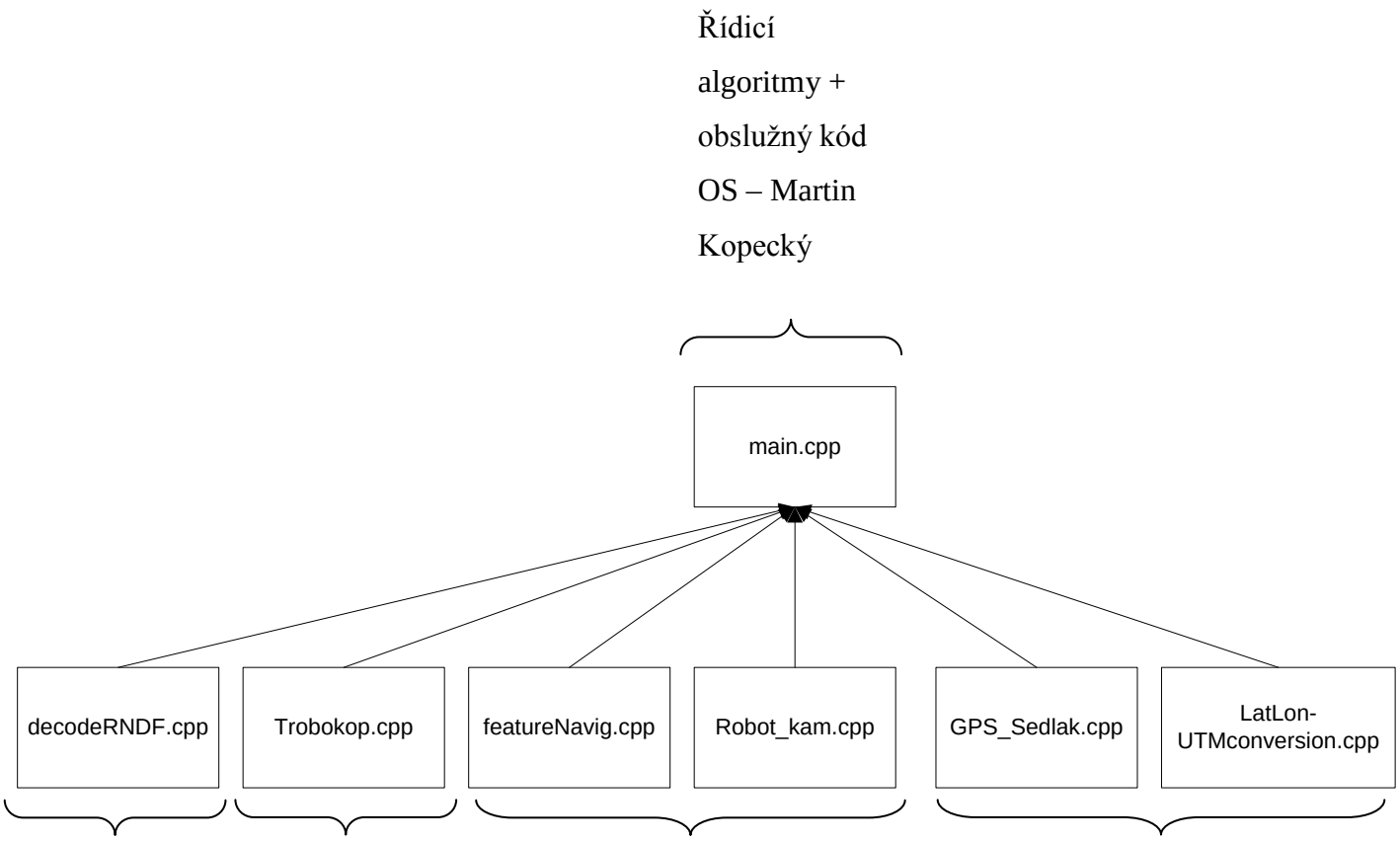
```
RNDF_name     "Luzankyv2.rnd"
num_segments  31
num_zones     0
format_version       1.0
creation_date 9/11/2009
segment       1
num_lanes     2
segment_name  A
lane  1.1
num_waypoints 4
exit  1.1.4   1.2.1
exit  1.1.4   12.1.1
exit  1.1.4   14.2.1
1.1.1 49.205993      16.607443
1.1.2 49.206172      16.607352
1.1.3 49.206548      16.607166
1.1.4 49.207002      16.606902
end_lane
lane  1.2
num_waypoints 4
exit  1.2.4   1.1.1
exit  1.2.4   18.2.1
exit  1.2.4   26.2.1
exit  1.2.4   30.1.1
1.2.1 49.207002      16.606902
1.2.2 49.206548      16.607166
1.2.3 49.206172      16.607352
1.2.4 49.205993      16.607443
end_lane
end_segment
segment       2
num_lanes     2
segment_name  B
lane  2.1
num_waypoints 2
exit  2.1.2   2.2.1
exit  2.1.2   4.2.1
exit  2.1.2   5.2.1
exit  2.1.2   17.1.1
2.1.1 49.205550      16.607609
2.1.2 49.205569      16.608543
end_lane
lane  2.2
num_waypoints 2
exit  2.2.2   2.1.1
exit  2.2.2   3.2.1
exit  2.2.2   18.1.1
exit  2.2.2   29.2.1
2.2.1 49.205569      16.608543
2.2.2 49.205550      16.607609
end_lane
end_segment
segment       3
num_lanes     2
segment_name  C
lane  3.1
num_waypoints 5
exit  3.1.5   2.1.1
exit  3.1.5   3.2.1
exit  3.1.5   18.1.1
exit  3.1.5   29.2.1
3.1.1 49.204884      16.607536
```

```
3.1.2 49.204989      16.607565
3.1.3 49.205111      16.607578
3.1.4 49.205449      16.607621
3.1.5 49.205550      16.607609
end_lane
lane  3.2
num_waypoints 5
exit  3.2.5   3.1.1
exit  3.2.5   4.1.1
exit  3.2.5   7.2.1
3.2.1 49.205550      16.607609
3.2.2 49.205449      16.607621
3.2.3 49.205111      16.607578
3.2.4 49.204989      16.607565
3.2.5 49.204884      16.607536
end_lane
end_segment
segment       4
num_lanes     2
segment_name  D
lane  4.1
num_waypoints 5
exit  4.1.5   2.2.1
exit  4.1.5   4.2.1
exit  4.1.5   5.2.1
exit  4.1.5   17.1.1
4.1.1 49.204884      16.607536
4.1.2 49.205066      16.607835
4.1.3 49.205341      16.608249
4.1.4 49.205460      16.608459
4.1.5 49.205569      16.608543
end_lane
lane  4.2
num_waypoints 5
exit  4.2.5   3.1.1
exit  4.2.5   4.1.1
exit  4.2.5   7.2.1
4.2.1 49.205569      16.608543
4.2.2 49.205460      16.608459
4.2.3 49.205341      16.608249
4.2.4 49.205066      16.607835
4.2.5 49.204884      16.607536
end_lane
end_segment
segment       5
num_lanes     2
segment_name  E
lane  5.1
num_waypoints 6
exit  5.1.6   2.2.1
exit  5.1.6   4.2.1
exit  5.1.6   5.2.1
exit  5.1.6   17.1.1
5.1.1 49.204998      16.610010
5.1.2 49.205040      16.609963
5.1.3 49.205256      16.609280
5.1.4 49.205435      16.608839
5.1.5 49.205546      16.608602
5.1.6 49.205569      16.608543
end_lane
lane  5.2
num_waypoints 6
exit  5.2.6   5.1.1
exit  5.2.6   6.1.1
```

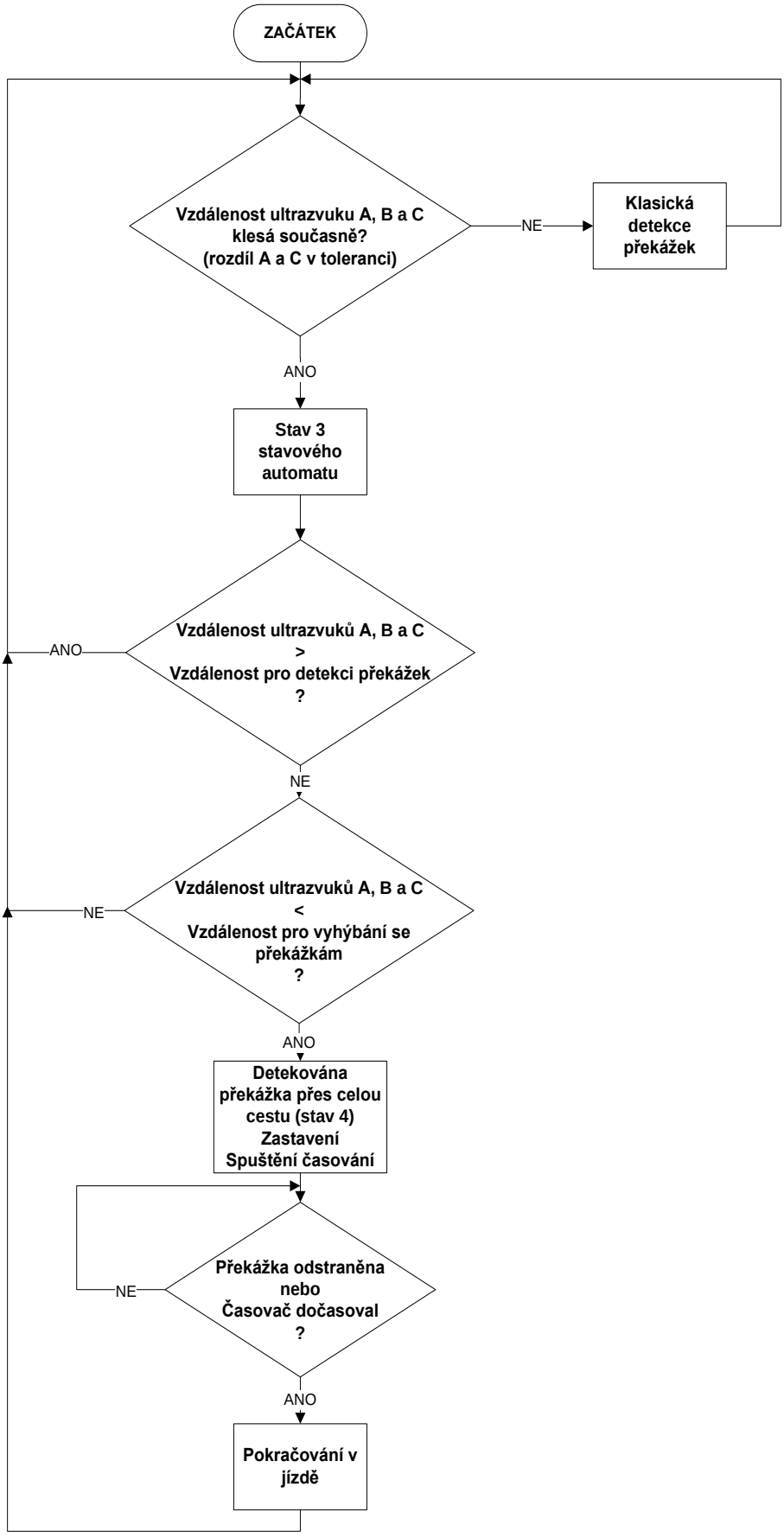
```
exit  5.2.6   19.1.1
5.2.1 49.205569       16.608543
5.2.2 49.205546       16.608602
5.2.3 49.205435       16.608839
5.2.4 49.205256       16.609280
5.2.5 49.205040       16.609963
5.2.6 49.204998       16.610010
end_lane
end_segment
segment       6
num_lanes     2
segment_name  F
lane  6.1
num_waypoints 5
exit  6.1.5   6.2.1
exit  6.1.5   7.1.1
exit  6.1.5   8.1.1
6.1.1 49.204998       16.610010
6.1.2 49.204845       16.609498
6.1.3 49.204629       16.608196
6.1.4 49.204661       16.607535
6.1.5 49.204703       16.607250
end_lane
lane  6.2
num_waypoints 5
exit  6.2.5   5.1.1
exit  6.2.5   6.1.1
exit  6.2.5   19.1.1
6.2.1 49.204703       16.607250
6.2.2 49.204661       16.607535
6.2.3 49.204629       16.608196
6.2.4 49.204845       16.609498
6.2.5 49.204998       16.610010
end_lane
end_segment
segment       7
num_lanes     2
segment_name  G
lane  7.1
num_waypoints 3
exit  7.1.3   3.1.1
exit  7.1.3   4.1.1
exit  7.1.3   7.2.1
7.1.1 49.204703       16.607250
7.1.2 49.204829       16.607431
7.1.3 49.204884       16.607536
end_lane
lane  7.2
num_waypoints 3
exit  7.2.3   6.2.1
exit  7.2.3   7.1.1
exit  7.2.3   8.1.1
7.2.1 49.204884       16.607536
7.2.2 49.204829       16.607431
7.2.3 49.204703       16.607250
end_lane
end_segment
...

end_file
```

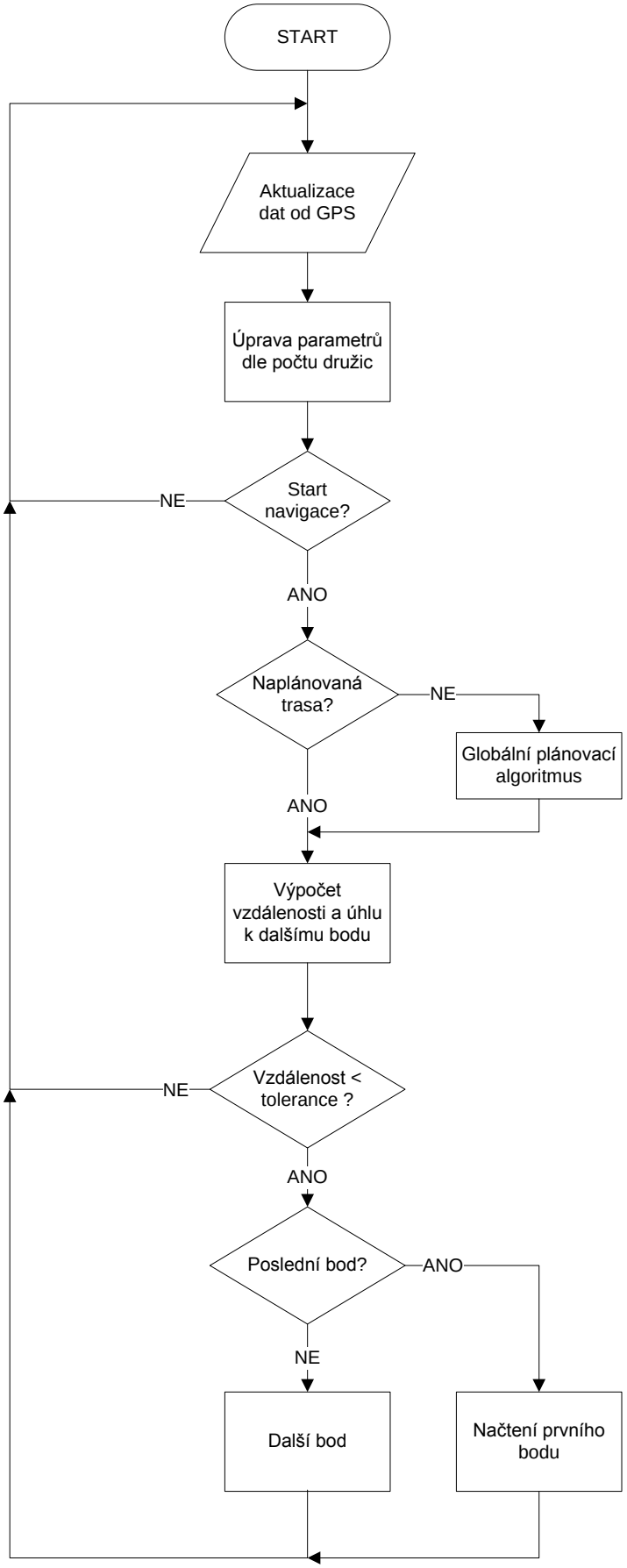
Příloha 2: Struktura souborů řídicího programu



Příloha 3: Vývojový diagram úpravy stavového automatu pro detekci překážek k detekování překážky přes celou cestu



Příloha 4: Vývojový diagram navigace robotu



Příloha 5: Vývojový diagram lokálního plánování